International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 12 (2018) pp. 10293-10297

An Optimized Approach to Achieve Energy Efficiency and Reliability Using Distance Aware Checkpointing Approach

Harpreet Kaur[1], Kamaljeet Kaur[2]

Computer Engineering and Technology, Guru Nanak Dev University, Amritsar, India.

Abstract

With the expanding interest and advantages of advanced computing infrastructure, real time computations can be performed on cloud framework. A real time framework can use intensive computing capacities and versatile virtualized condition of cloud environment to execute jobs of real time classification. In most real time applications, preparing is done on remote cloud computing centres. So it becomes prone to errors, because of the undetermined idleness and free control over controlling hub. On the opposite side, the large community of the continuous frameworks are likewise security basic and ought to be exceedingly solid. So there is an expanded necessity for fault resistance to accomplish dependability for real time cloud Infrastructure. In this paper, checkpointing with minimum distance handling mechanism framework is used for virtualized specialist co-ops and fault recovery for cloud computing. By using the checkpoints the task can be recover even though our system crashed with less chances of data loss. For recovery purposes shortest path algorithm with minimum distance vector metric is used to reduce the time required to produce backup. This paper presents a running usage of this infrastructure and its evaluation, exhibiting that it is a viable method to make speedier checkpoints with low impedance on assignment execution and proficient undertaking recovery after a node failure.

Keywords: Cloud Computing, Checkpointing, distance, disk usage, Fault tolerance.

INTRODUCTION

The term cloud, is utilized as a part of this paper, seems to have its causes in arrange outlines that discusses about the web, or different parts of it, as schematic clouds.

Cloud computing isn't something that all of a sudden seemed overnight; in some frame it might follow back to a period when PC frameworks remotely time-shared registering assets and applications [1]. Cloud computing refers to the various sorts of services and applications being conveyed in the web cloud, and the way that, much of the time, the gadgets used to get to these services and applications don't require any unique applications. Many organizations are conveying services from the cloud. Some outstanding cases starting at 2010 incorporate the accompanying: Google is a private cloud that it utilizes for conveying a wide range of services to its clients, including email, record applications, content interpretations, maps, web examination, and considerably more [2].

Failures and outages influenced most famous distributed computing suppliers, for example, Samsung, Microsoft trade, Verizon Wireless and others. Enhanced FT techniques can altogether improve the unwavering quality of cloud frameworks. As computing in cloud is performed at remote frameworks there are more conceivable outcomes of failures because of undetermined idleness, and loss of control over registering hub. Consequently, it is fundamental that remote frameworks are exceedingly solid. There is an absence of concentrates on Fault Tolerance in Cloud Computing (FTCC) systems [3]. The large community of the present work just spotlights on the definitions and present a few procedures, or examines toolbox that gives models to upgrade the unwavering quality of the framework. Our examination endeavours to break down the accessible fault resistance strategies and models in distributed computing. We abridge the FTCC strategy and models in two unique tables to show their qualities and shortcoming [4].

The way toward taking a depiction of the present condition of the running application on to a steady stockpiling is called checkpoint. This is the most usually utilized strategy for fault resistance. At the point when a fault is experienced the application can be restarted from the most recent checkpoint state. This diminishes the recompilation time quantifiably. Checkpoint arrangement depends on a few characteristics are spoken to given underneath:

A. Deliberation Level:

The level of deliberation at which the present condition of the application is spared is the criteria for grouping. Under this arrangement there are three writes

Framework Level Checkpoint: Automatic and Transparent registration of utilizations at the working framework or middleware level is given utilizing this strategy. This system has no learning about the qualities of the application [5].

1. The entire procedure picture of the application is caught. It includes the characteristics of process state, for example, program counter, registers and memory saved money on the steady stockpiling.
2. User or Application Level Checkpoint: Fault Tolerance is accomplished by the application inside itself by giving self containing code. The application is composed in such a way it restarts naturally utilizing the data in the restart document.
3. Blended Level Checkpoint: It is the mix of System Level Checkpoint and User Level Checkpoint.

B. Message Coordination:

The way of framework deals with the in-travel and vagrant messages is the criteria for this arrangement. Under this head, the accompanying are sorts.

1. **Uncoordinated Checkpointing:** Each procedure of the application takes the checkpoint autonomously without organizing with alternate process. Absence of synchronization makes the focuses conflicting and amid rollback, the focuses must be hunt down predictable worldwide checkpoint [6].

Benefits:

1. Control message trade is kept away from.

2. Process can perform checkpoint exclusively.

Demerits:

1. Probability of Domino impact.

2. Numerous checkpoints for a procedure may prompt stockpiling overhead.

3. A procedure may take a registration that need not ever add to a steady worldwide registration.

2. Synchronous/Coordinated Checkpointing: In this approach, the procedures keep up the predictable worldwide checkpoint. It takes after two stages confer. The speculative checkpoints taken in the primary stage are made changeless in the second stage. On fault, the procedures will move back to the lasting checkpoint.

Benefits:

1. Single lasting checkpoint brings down the steady stockpiling overhead.

2. Does not experience the ill effects of domino impact

3. Straightforward rollback method

Demerits:

1. It includes trade of different correspondence on trading messages.

There are two kinds of facilitated registration

1. Blocking Check-indicating: forestall vagrant messages, the procedure stays obstructed, until the whole checkpointing movement is finished in the wake of taking a neighbourhood checkpoint. The procedure is permitted to continue its execution when it completes its nearby checkpoint. The hindrance is the calculation is obstructed amid the registration [7].

2. Non-blocking Check-pointing: The in travel and vagrant messages may exist at the season of neighbourhood checkpoint. The procedures require not stop their execution while taking checkpoints. Keeping a procedure from accepting an application message that would bring about conflicting checkpoint in one of the issues experienced in this write [8].

3. Correspondence Induced/Hybrid/Quasi-synchronous Checkpoint: This strategy authorizes at the production of worldwide checkpoint that is ungraceful. The neighbourhood checkpoints are made autonomously. Notwithstanding, domino impact is kept away from by constraining extra checkpoints in order to guarantee the inevitable advance of the worldwide checkpoint [2].

LITERATURE SURVEY

Checkpointing, i.e., recording the unstable condition of a virtual machine (VM) running as a visitor in a Virtual Machine Monitor (VMM) for later rebuilding, incorporates putting away the memory accessible to the VM. Regularly, a full picture of the VM's memory alongside processor and gadget states are recorded. With visitor memory sizes of up to a few gigabytes, the span of the checkpoint pictures turns out to be increasingly of a worry. In this work we introduce a procedure for quick and space-productive checkpointing of virtual machines. As opposed to existing strategies, our system kills repetitive information and stores just a subset of the VM's memory pages. Our procedure straightforwardly tracks I/O tasks of the visitor to outside capacity and keeps up a rundown of memory pages whose substance are copied on non-unpredictable storage. At a checkpoint, these pages are barred from the checkpoint picture. We have actualized the proposed procedure for virtualized and in addition completely virtualized visitors in the Xen VMM. Our examinations with a virtualized visitor (Linux) and two completely virtualized visitors (Linux, Windows) demonstrate a huge lessening in the span of the checkpoint picture and additionally the time required to finish the checkpoint. Contrasted with the current Xen implementation, we accomplish, by and large, an 81% decrease in the put away information and a 74% diminishment in the time required to take a checkpoint for the virtualized Linux visitor. In a completely virtualized condition running Windows and Linux visitors, we accomplish a 64% diminishment of the picture measure alongside a 62% decrease in checkpointing time [9].

This paper introduces an approach for giving high accessibility to the solicitations of cloud's customers. To accomplish this target, failover methodologies for distributed computing utilizing incorporated checkpointing calculations are purposed in this paper. Purposed system coordinate checkpointing highlight with stack adjusting calculations and furthermore influence multilevel checkpoint to diminishing to checkpointing overheads. For execution of purposed failover systems, a cloud recreation condition is created, which can give high accessibility to customers if there should arise an occurrence of failure/recovery of administration hubs. Likewise in this paper examination of created test system is made with existing techniques. The purposed failover procedure will chip away at application layer and give profoundly accessibility to Platform as a Service (PaaS) highlight of distributed computing [10].

Distributed computing is turning into an essential answer for giving adaptable processing assets by means of Internet. Since there are a huge number of hubs in data centre, the likelihood of server failures is nontrivial. In this way, it is a basic test to

ensure the administration unwavering quality. Fault-tolerance systems, for example, checkpoint, are generally utilized. Due to the failure of the edge switches, the checkpoint picture may end up difficult to reach. Along these lines, current checkpoint-based fault resilience technique can-not accomplish the best impact. In this paper, we propose an ideal checkpoint strategy with edge switch failure-mindful. The edge switch failure-mindful checkpoint technique incorporates two calculations. The main calculation utilizes the data centre topology and correspondence characteristic for checkpoint picture stockpiling server determination. The second calculation utilizes the checkpoint picture stockpiling trademark and in addition the data centre topology to choose the recovery server. Recreation tests are performed to exhibit the viability of the proposed technique [11].

Checkpointing with rollback recovery is an entrenched strategy to endure transient faults. In any case, it causes critical time and vitality overheads, which go squandered in without fault execution states and may not be achievable in hard ongoing frameworks. This paper shows a low-overhead two-state checkpointing (TsCp) plot for fault-tolerant hard constant frameworks. It separates between the fault and faulty execution states and use two kinds of checkpoint interims for these two distinct states. The primary kind is non uniform interims that are utilized while no fault has happened. These interims are resolved in light of deferring checkpoint additions in sans fault states, with the point of diminishing the quantity of checkpoint inclusions. The second sort is uniform interims that are utilized from the time when the principal fault happens. They are resolved to limit execution time for faulty states, leaving additional time accessible for vitality administration in without fault states. Trial assessment on an installed processor (LEON3) and a rising nonvolatile memory innovation (ReRAM) outlines that TsCp altogether lessens the quantity of checkpoints (62% by and large) contrasted and past works, while safeguarding fault resistance. This outcomes in 14% and 13% lessened execution time and vitality utilization, separately. Besides, we join TsCp with dynamic voltage scaling (DVS) and accomplish up to 26% (21% by and large) vitality sparing contrasted and the cutting edge strategies [12].

RESEARCH GAP

The existing literature uses two algorithms for optimizing the resource utilization and enhances the reliability. Reliability indicates that task submitted to the VM is successfully completed by the virtual machine. The first approach followed by the existing literature is to select the server optimally. To achieve this, storage server which belongs to same subnet (virtual machine belonging to same host) from where back up is to be initiated, is selected to store checkpoint image. In the second algorithm, recovery server is selected. Recovery server is selected on the basis of observing the request initiation. If it is from the same pod then server is selected from the same pod for recovery. VM is restarted and process is again started at VM chosen for holding server backup.

PROBLEM DEFINITION

The recovery image storage mechanism can be accommodated with shortest path selection strategy to store the checkpoint image at nearest server available. As the nearest server is selected for storage time required for recovery is considerably reduced. Thereby energy consumption also reduces.

In the second algorithm, recovery server selection process does not consider the distance metric. Again shortest path algorithm can be used to reduce the time required to produce backup. Hence accommodating distance within these two algorithms can reduce execution time and energy consumption. Since distance is directly proportional to energy consumption.

PROPOSED METHODOLOGY

Algorithm 1:

Storage server selection Input: service providing server Ps, the subnet subs that Ps belongs to, the pod pods that Ps belongs to output: checkpoint image storage server Tserver

1. Add all the servers to PM_List_i categorised by pod
2. Check for the distance between the checkpoint image server to the backup server
3. $Min=distance_i$
4. For i=1: server_in_same_pod
   If($Min>distance_{i+1}$)
   $Min=distance_{i+1}$
   End of if
   End of loop
5. Select Server[Min] distance for backup
6. Eliminate Server[Min] from PM_List
7. Repeat step 1 to 6 for all the PMs in the list

Algorithm 2

Recovery server selection in case of failure

1. Check for the distance between VM failed and recovery server to identify the pod
2. Select Server[min] for recovery
3. Restart the VM
4. Execute the job
5. Produce result in terms of resource utilization and execution time.

Fault Injection and Recovery- In this paper response fault will be inject so that during transmission of the problems that are occurring will be easily resolved. Then by applying the above algorithm we will recover the faults that are occurring in the system by selecting the best server that can complete the jobs immediately that was stuck due to fault occurring in the system.

RESULTS

Execution time- It is the total time required by the processor to complete the job or accomplish the task. Execution time will be considered in terms of existing and proposed paper.

Table 1: In terms of execution time

Cloudlets	Execution Time in ms(Existing)		Execution Time in ms(Proposed)	
	Before fault	After fault	Before fault	After fault
50	2394	2156	1589	1325
100	2189	1987	1756	1633
150	2009	1982	1673	1542
200	1998	1876	1527	1237
250	2135	2097	1865	1652

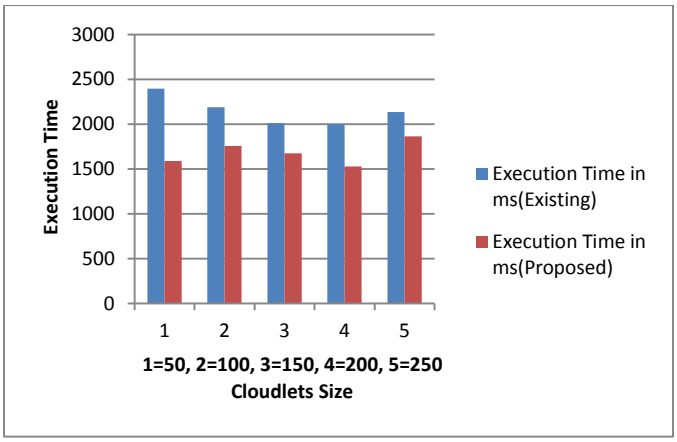

Figure 1. Plot in terms of execution time

Down Time: It is the total time during which a machine is out of action or not available for usage.

Table 2: In terms of down time

Cloudlets	Down Time in ms(Existing)		Down Time in ms(Proposed)	
	Before fault	After fault	Before fault	After fault
50	1985	1732	1234	1152
100	1999	1865	1339	1234
150	2072	1987	1432	1345
200	1854	1653	1267	1098
250	1863	1735	1532	1432

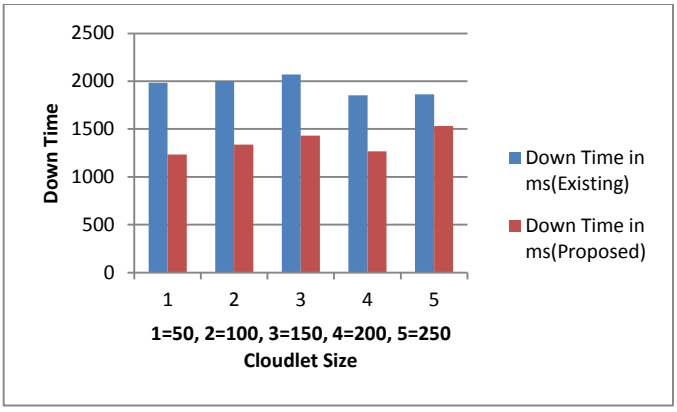

Figure 2. Plot down time of proposed and existing paper

Disk Usage- It is the total space used by the process in the computer. The disk usage of proposed and existing paper will be considered in terms of Mega bytes.

Table 3: In terms of disk usage

Cloudlets	Disk Usage in MB(Existing)		Disk Usage in MB(Proposed)	
	Before fault	After fault	Before fault	After fault
50	2658	2354	1569	1345
100	2344	2134	1855	1678
150	2268	2093	1876	1765
200	2678	2456	1985	1876
250	2174	1987	1865	1567

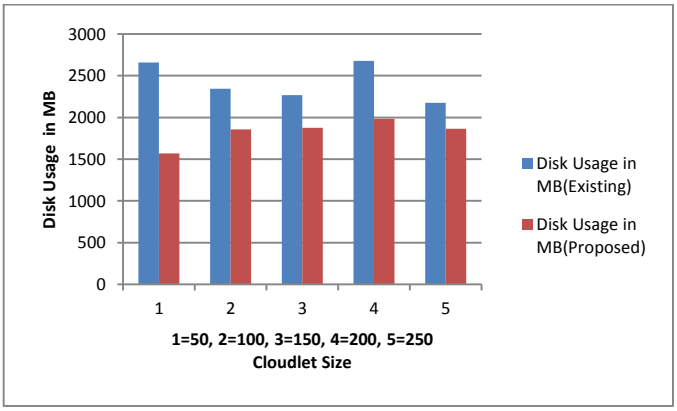

Figure 3. Plot in terms of disk usage

CONCLUSION AND FUTURE SCOPE

In this paper we are examining the problem of cloud service reliability. In our proposed approach checkpointing technique is used in cloud computing to recover the data which can be lost due to system crash or many more reasons. To overcome this problem we use shortest path distance algorithm so that task during data recovery can be migrated to the minimum distance servers. So that it can be easily available or accessible to obtain optimal results in terms of disk usage, image size, total execution time, average lost time etc. In

future we can collaborate multiheuristic approach for reliability enhancement.

REFERENCES

[1] Belalem, G. & Limam, S., 2011. Fault Tolerant Architecture to Cloud Computing Using Adaptive Checkpoint", ACM International Journal of Computer Applications and computing, vol 1, no. 4, pp. 60-69, 2011.

[2] J. Ansel, K. Arya, and G. Cooperman, "DMTCP: Transparent checkpointing for cluster computations and the desktop," Proc. IEEE Int. Parallel Distrib. Process. Symp.(IPDPS), 2009.

[3] Pandey, P., Dhasal, P. & Pandit, "Implementation of RSA RC5 Algorithm in Cloud", (IJCSIT) International Journal of Computer Science and Information Technologies,, vol. 6, no. 1, pp.224–227, 2015.

[4] J. Broberg, "Dynamic Fault-tolerance and Mobility Provisioning for Services on Mobile Cloud Platforms," 5th IEEE international conference on Mobile Cloud, services and Engineering, pp. 131-138, 2017.

[5] El-Sayed, N. & Schroeder, "To checkpoint or not to checkpoint: Understanding energy-performance-I/O tradeoffs in HPC checkpointing", IEEE International Conference on Cluster Computing, pp.93–102,2014.

[6] Bautista Gomez, A. Nukada, N. Maruyama, F. Cappello, and S. Matsuoka,, "Low-overhead diskless checkpoint for hybrid computing systems", IEEE 17th International Conference on High Performance Computing, 2010.

[7] M. E. M. Diouri, O. Glück, L. Lefevre, and F. Cappello, "Energy considerations in checkpointing and fault tolerance protocols," in Proc. IEEE International Conference Dependable Systems and Networks workshops, pp. 1–6, 2012.

[8] Q. Jiang, Y. Luo, and D. Manivannan, "An optimistic checkpointing and selective message logging approach for consistent global checkpoint collection in distributed systems," IEEE International Parallel and Distributed Processing symposium, pp. 1-10, 2007.

[9] E. Park, B. Egger, and J. Lee, "Fast and Space-Efficient Virtual Machine Checkpointing, Proceedings of the 7th ACM SIGPLAN/SIGOPS international conference on Virtual execution environment, vol. 46, no. 7, pp. 75–86, 2011.

[10] Singh, D., Singh, J. & Chhabra, "High availability of clouds: Failover strategies for cloud computing using integrated checkpointing algorithms" IEEE Proceedings International Conference on Communication Systems and Network Technologies, CSNT 2012, pp.698–703, 2012.

[11] A. Zhou, Q. Sun, and J. Li, "Enhancing Reliability via Checkpointing in Cloud Computing Systems," IEEE China Communications, vol. 14, no. 7, pp. 1-10, 2017.

[12] M. Salehi, M. K. Tavana, S. Rehman, M. Shafique, A. Ejlali, and J. Henkel, "Two-State Checkpointing for Energy-Efficient Fault Tolerance in Hard Real-Time Systems," IEEE Trans. Very Large Scale Integr. Syst., vol. 24, no. 7, pp. 2426–2437, 2016.