Developing precise and efficient Metrics for Adaptive Checkpoint Scheduling of Virtual Machine Replication on Multi-core

Singh Angad (3035124764) FYP Advisor : Dr Heming Cui

I. **Introduction**

Today, there is a growing demand for online services deployment on virtualized infrastructures [1]. Online services are processing more and more requests concurrently nowadays and that requires virtual machines (VM) to utilize and more and more virtual CPU's on multi-core hardware. Because of this rise in cloud computing, hardware failures have become more common [5]. This implies that the need for high availably is rising.

However, high availability is hard to obtain. Previously, high availably was only possible using commercial hardware or application specific replication [2]. Several Solutions have been given in the past to make high availability common place and make the Virtual Machines more fault tolerant. One of the most common solution is check point recovery. The entire state of the Virtual machine is copied at very high frequency's and the changes are propagated to the backup virtual machine almost instantaneously. While the changes are being copied, the virtual machine is paused and no output is being released to the user. The output is released when the copy is made successfully. This process is carried out at fixed intervals [5].

The problem with this method is that it can cause significant overhead because of the large amount of data that needs to be copied and transferred, even on uni-processor VM setups [3]. There have been research studies on adaptive checkpointing (eg. [3], [4]) but this is a vast field and the process of adaptive checkpointing can be made more efficient to improve user experience and fault tolerance.

I intend to investigate further in this area, and come up with metrics and algorithms for adaptive checkpointing to decrease the overhead, especially on multi-core systems.

The rest of my plan is organized as follows. Section II covers the background of the project including a literature review of three research papers related to this project. Section III describes the objective of the project. Section IV discusses the intended methodology of the project. Section V gives a detailed schedule for the project. And Section IV concludes.

II. **Background**

A. Literature Review

Three papers have been discussed here related to this project.

1. “Remus: High Availability via Asynchronous Virtual Machine Replication” [2]

Remus [2] adopts a frequent checkpoint backup model to maintain Haigh Availability. Remus keeps a backup host and propagates the changes to this backup host from the primary at a predefined interval.

The network output from Remus is stored in the buffer until the system state synchronizes with the backup’s state. The checkpoint is very frequent, at every 25 milliseconds [2]. Disk changes are propagated to the backup asynchronously since the entire disk snapshot needs to be transferred. The output is only released once the two machines have synchronized and a confirmation from the backup machine has been received. This essentially takes care of the output commit problem [2] which means that any system state which has been displayed should be able to be recoverable [6]. When a failover occurs, the backup is started and replaces the primary. However, it is important to note that the virtual machine does not do any execution till a failover occurs [2].

This approach leads to a significant overhead at times because of the huge amount of data which needs to be transferred.

2. “Workload Adaptive Checkpoint Scheduling of Virtual Machine Replication” [3]

This paper discusses adaptive checkpointing based on an analysis of workloads of several applications. This paper attempts to reduce the overhead by dynamically adjusting the checkpoint frequency based on properties such as the number of dirtied memory pages, the number of disk I/O operations, the number of transferred network packets and the network bandwidth available for replication [3].

However this paper does not provide solutions which exploit the availability of extra CPU cores. This paper also did not take into the fact that there are unnecessary program idle slots in between two checkpoints [1].

3. “Adaptive Remus: adaptive checkpointing for Xen-based virtual machine replication” [4]

This paper attempts to decrease the overhead by adapting the checkpoint frequency according to the application being run on it. It suggests the virtual machine to run in two modes. (i) network mode: where it increases the checkpoint frequency where high output traffic is detected. (ii) processing mode: where it decreases the checkpoint frequency because of the low output traffic [4].

This paper also doesn't investigate the two areas mentioned above.

B. Checkpoint Frequency

Checkpoint frequency is the time interval at which the virtual machine would be stopped and the changes would be propagated to the backup machine. Due to the amount of data which needs to be transferred this task can prove to create a significant overhead even on uni-processors [3]. On multi-core systems the overhead is even more. To reduce the overhead, adaptive checkpointing needs to be implemented.

This area is still very lucrative and more metrics and algorithms can be developed to significantly improve the performance of Remus.

III. Project Objective

Research has suggested that dynamic adaptive checkpointing tends to be better and faster than fixed frequency checkpointing [3,4]. My goal is to build on this research and improve the performance even further.

My main aim is to come up with metrics and algorithms for adaptive checkpointing to decrease the overhead, especially on multi-core systems.

The primary goal would be to analyze several applications on multi core system using Remus [2] and Adaptive Remus [4] and observe the results and look for areas for improvements. Properties like the number of dirty pages which need to be transferred, the similarity between the pages being transferred, redundancy in the transferring process, disk write operations, network I/O operations etc. all effect the performance of the system. Identifying key patterns may lead to improved performance.

I would further investigate into using extra CPU cores which are not being used by the system to leverage in my solution for making my system use less time and decrease the overhead. Something similar has been done in PLOVER [1] where dirty page hashes are calculated concurrently by using the same number of threads as the number of CPU cores in the virtual system. Doing this reduces the time taken to compute hashes and this process effectively doesn't become a bottleneck [1].

In between checkpoints, there are a lot server idle status when server programs finish processing the current requests and clients' TCP windows are full [1]. I will investigate into using this fact and leveraging it to develop a metric and further enhancing the performance.

I would further research on using a lightweight compression algorithm to reduce the amount of data being transferred and further improving the performance.

The final deliverables would be a set of metrics and algorithms for adaptive checkpoint scheduling and a running system to test on.

IV. Project Methodology

The first goal is to conduct tests with Remus by running application on multi-core systems. Then these results would be evaluated and areas for improvement will be detected. The second step would to be develop several potential metrics and algorithms which can be used to manipulate the checkpoint frequency to improve the performance.

The next step would be to implement these algorithms and metrics and integrate them in Remus. Then the testing will be done on a 24-core system. A comparison will be made with other systems such as PLOVER [1] and other Remus based softwares[3,4].

V. Project Schedule and Milestones

This is the tentative schedule for the project.

Analysis of the existing softwares	Oct 2017
Invent/ Explore relevant metrics and algorithms	Nov 2017 - Dec 2017
Implementation with Remus	Jan 2017
Experimentation	Feb 2017
Results Analysis and Comparison	Mar 2017
Completion	Apr 2017

VI. Conclusion

In this project I intend to improve the performance of a virtual machine running while actively maintaining a backup using Remus. This will be achieved by coming up with precise and efficient metrics and algorithms for adaptive checkpoint scheduling on the virtual machine. Several approaches have been suggested in section III and other approaches will also be considered based on the initial analysis'.

VII. Resources

[1] PLOVER: Fast and Scalable Virtual Machine Fault-tolerance on Multi-core.

[2] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A. Warfield, Remus: High availability via asynchronous virtual machine replication, in Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, NSDI'08, USENIX Association, Berkeley, CA, USA, 2008, pp. 161–174

[3] B. Gerofi and Y. Ishikawa, Workload adaptive checkpoint scheduling of virtual machine replication, in IEEE 17th Pacific Rim International Symposium on Dependable Computing (PRDC), Pasadena, CA, December 2011, pp. 204–213.Adaptive Remus

[4] Which Hardware Fails the Most and Why. http://www.storagecraft.com/blog/hardwarefailure/

[5] Strom, R.E. and Bacon, D.F. and Yemini, S.A., "Volatile logging in n-fault-tolerant distributed systems," in Fault-Tolerant Computing, Eighteenth International Symposium on, Jun 1988, pp. 44 –49.