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

Interim Report, Final Year Project 2017-18

Updated on 20 January 2018

Version 1.2

Angad Singh (3035124764)
Project Advisor: Dr Heming Cui
Abstract

High availability, that is the ability to recover quickly after a hardware failure, is hard to achieve yet desirable. Nowadays, high availability is achieved through virtual machines using a checkpoint mechanism where changes are propagated to a backup machine at a fixed checkpoint (up to 40 times in a second). This can produce unnecessary load on the system especially multicore systems. Previous work has shown that adaptive checkpointing is a more efficient solution when modeled around the properties of the software being run on it. The objective of this project is to produce relevant metrics and algorithms for adaptive checkpointing which can be integrated into existing solutions such as PLOVER, REMUS etc. The project will be carried out in 3 phases. The first phase would consist of experiments with popular softwares using PLOVER and other phases would focus on utilizing these results and observations to increase system performance. Interesting observations have arisen from the initial experiments which can potentially lead to better system performance. Several other experiments and research have been planned and scheduled.

Acknowledgment

I would like to extend my gratitude to my FYP advisor, Dr Heming Cui, for helping me with the planning and execution of the project, Cheng Wang for providing me all the support and resources I needed and Cezar Cazan for providing valuable feedback on my work and helping me improve the quality of my work.
# Table of Contents

Abstract 2  
Acknowledgment 2  
Table of Contents 3  
List of Tables 3  
Abbreviations 4  
I. Introduction 4  
II. Background 5  
III. Literature Review 6  
   i. Remus: High Availability via Asynchronous Virtual Machine Replication 6  
   ii. PLOVER: Fast and Scalable Virtual Machine Fault Tolerance on Multi-core 6  
   iii. Workload Adaptive Checkpoint Scheduling of Virtual Machine Replication 6  
IV. Objective 7  
V. Methodology 7  
   i. Phase 1 8  
   ii. Phase 2 8  
   iii. Phase 3 8  
VII. Project Status 8  
VIII. Future Plans 9  
IX. Conclusion 9  
X. References 9
List of Tables

1 The status of all the tasks associated with the project along with their projected date of completion. 8

Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output</td>
</tr>
<tr>
<td>GB</td>
<td>GigaByte</td>
</tr>
<tr>
<td>SSD</td>
<td>Solid State Drive</td>
</tr>
<tr>
<td>GHZ</td>
<td>Gigahertz</td>
</tr>
<tr>
<td>TB</td>
<td>Terabyte</td>
</tr>
<tr>
<td>GBPS</td>
<td>GigaBytes per second</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>RDMA</td>
<td>Remote Directory Memory Access</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>PHP</td>
<td>Hypertext Preprocessor</td>
</tr>
</tbody>
</table>
I. Introduction

Recently, there has been an increasing 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 more and more virtual CPUs 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 availability is rising.

However, high availability is hard to obtain. Previously, high availability was only possible using commercial hardware or application specific replication softwares[2]. Several Solutions have been given in the past to make high availability common place. One of the most common solutions is checkpoint recovery. The entire state of the Virtual machine is copied at very high frequencies and the changes are propagated to the backup virtual machine almost instantaneously. While the changes are being copied, the virtual machine runs speculatively and no output is being released to the user. The output is released when the two machines have been synchronized 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 uniprocessor VM setups, so frequently [3]. Research (eg [3], [4]) has suggested that adaptive checkpointing instead of fixed frequency checkpointing can improve the system performance.

The aim of this project is to further investigate in this area, and propose metrics and algorithms for adaptive checkpointing to decrease the overhead, especially on multi-core systems. The final deliverable will be a set of metrics and algorithms for adaptive checkpointing. These metrics and algorithms would be written in pseudo code and could potentially be integrated with many of the existing checkpointing solutions such as PLOVER, REMUS etc. to improve their performance.

The rest of this report is organized as follows. Section II covers the background of the project. Section III covers Literature Review of three papers related to this field. Section IV describes the detailed objective of the project. Section V describes the proposed methodology for the project. Section VI details and discusses the interim results. Section VII talks about the project status and the problems encountered during the project. Section VIII discusses the future plans for the project and section IX concludes this report.

II. Background

High Availability is defined as the ability of a system to be operational and fully functional for a suitable length of time [8]. To achieve High Availability we need to make sure that there is a backup for the system which can be accessed any time without any major delay in case of hardware failure and the web service can be resumed from the point where the system crashed. There should ideally be no data or network packet lost. This can be achieved by using commercial hardware built for this purpose or restructuring the code of each and every application in the virtual machine to include complicated logic for recovery [2]. However, these solutions are expensive and are not available to the masses. Another approach would be to
propagate the system state synchronously to a backup machine, but this will slow down the system and the systems memory throughput would be comparable to a replication performing network device, which is not desirable.

Remus [2] provided a very good solution to this problem by replicating the system state at frequent predefined checkpoints. It lets the host system perform its computations in speculative state and propagates the changes asynchronously to the backup machine at every 25 ms [2]. Several new Systems such as PLOVER also adopt a similar fixed checkpoint frequency model [1].

Another problem which has to be tackled while working with backing up virtual machines is the output commit problem. The output commit problem is that a system cannot send data to the outside world until and unless, it has verified that the data is error free and has been backed up [9]. In case of a hardware failure the end user should not see any data loss.

III. Literature Review


Remus [2] adopts a frequent checkpoint backup model to maintain High 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] (Which implies a rate of upto 40 checkpoints in a second). 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 until a failover occurs [2].

Due to the amount of data which needs to be transferred this task can prove to create a significant overhead even on uniprocessors [3]. On multi-core systems the overhead is even more because of more number of cores working concurrently, hence increasing the workload per time unit.


State machine replication (SMR) enforces the same total order of inputs for a service which is replicated along hosts. By doing this most of the memory pages in the two hosts get updated and thus do not require to be transferred from the primary to the secondary host.

This paper discusses the system PLOVER which is the first Virtualized State machine Replication (VSMR) System. VSMR enforces the same total order of inputs across replicated
virtual machines. This keeps majority of the memory pages updated and very few divergent pages have to be transferred across systems. Plover uses an adaptive checkpoint frequency model which tends to avoid unnecessary program idle slots.


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].

Apart from the adaptive checkpoint scheduling, the paper also implements a fine grained copy-on-write mechanism to avoid the downtime caused by checkpointing [3]. This has been achieved by only locking the memory pages of which the values have to be transferred to the backup machine, allowing the virtual machine to run concurrently [3].


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 network traffic is detected. (ii) Processing mode: where it decreases the checkpoint frequency because of the low output network traffic [4].

IV. Objective

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

The main aim of this project is to produce metrics and algorithms for adaptive checkpointing to decrease the overhead, especially on multi-core systems and effectively improve the system performance.

The final deliverable will be a set of metrics and algorithms for adaptive checkpointing. These metrics and algorithms would be written in pseudo code and could potentially be integrated with many of the existing checkpointing solutions to improve their performance.

V. Methodology

The project will be carried out in three phases. The first phase consists of mainly experiments with current available solutions and finding out areas to improve them. The second phase will focus on using the knowledge obtained from the previous phase and developing metrics and algorithms based on that. The final phase will focus on testing and evaluation.
All the tests and experiments will be done on Dell R430 servers with Linux 3.16.0, 2.6 GHz Intel Xeon CPU with 24 hyper-threading cores, 64GB memory, and 1TB SSD. All NICs are Mellanox ConnectX-3 Pro 40Gbps connected with infiniband [10]. The ping latency between every two replicas is 84 microseconds (the TCP/IP over RDMA round-trip latency).

i. Phase 1

The first goal is to conduct tests with Plover [1] and Remus [2] by running several popular applications and collecting relevant properties. Then these results would be evaluated to find areas for improvement.

Seven programs have been selected for the tests. These are not only widely used but also are diverse in terms of network traffic and processing. The tests will be conducted on Django Content Management System (which consists of Nging, Python and MySQL), Apache (using PHP), Tomcat, FTP (File Transfer Protocol) server, Hadoop, Kernel Compilation and NAS Parallel Benchmarks.

The properties being studied are:– 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. These properties have been selected because they account for most of the avoidable overhead in the current solutions.

ii. Phase 2

This phase would focus on evaluating the results obtained from the previous phase and finding areas for improvement. Based on these findings, algorithms and metrics for adaptive checkpointing will be created.

iii. Phase 3

The final phase of this project will involve implementing these algorithms in Plover followed by rigorous testing and evaluation. Similar tests as in phase 1 will be conducted to measure the success of this project.

VII. Project Status

Currently the first and the second phase of the project are underway. Until now, tests have been conducted on several softwares. Several interesting observations have shown up. Network intensive applications require significantly more checkpoints as compared to processing intensive applications since they involve more output traffic. The number of dirty pages tends to increase fast in the beginning of the epoch but the growth rate reduces over time. Research on using results from these experiments is ongoing.
<table>
<thead>
<tr>
<th>Task</th>
<th>Scheduled Completion</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background research and literature review</td>
<td>Sep 2017 - Oct 2017</td>
<td>Finished</td>
</tr>
<tr>
<td>Analysis of the existing softwares (Phase 1)</td>
<td>Nov 2017 - Dec 2017</td>
<td>In-Progress</td>
</tr>
<tr>
<td>Invent/ Explore relevant metrics and algorithms (Phase 2)</td>
<td>Jan 2017</td>
<td>In-Progress</td>
</tr>
<tr>
<td>Implementation with Plover (Phase 3)</td>
<td>Feb 2018</td>
<td>-</td>
</tr>
<tr>
<td>Experimentation (Phase 3)</td>
<td>Feb 2018</td>
<td>-</td>
</tr>
<tr>
<td>Results Analysis and Comparison (Phase 3)</td>
<td>Mar 2018</td>
<td>-</td>
</tr>
<tr>
<td>Completion</td>
<td>Apr 2018</td>
<td>-</td>
</tr>
</tbody>
</table>

*Table 1: The status of all the tasks associated with the project along with their projected date of completion.*

**VIII. Future Plans**

The primary goal is to continue collecting more data using the rest of the four programs. As part of phase 2, after the data has been collected from the experiments, work would be done on developing appropriate metrics and algorithms to reduce the overhead.

**IX. Conclusion**

The intention of this project is to improve the performance of a virtual machine running while actively maintaining a backup using Plover. This will be achieved by coming up with precise and efficient metrics and algorithms for checkpoint scheduling on the virtual machine. Several approaches, such as adaptive checkpointing, memory compression and idle state utilization, have been suggested in sections V and VIII and other approaches will also be considered based on the initial analysis. Some initial experiments have already been carried out and certain interesting observations have arisen. Decreasing the checkpoint frequency tends to decrease the number of memory pages being transferred. These observations have the potential to reduce the overhead on multi core systems and enhance the system performance. Further experiments have also been planned and scheduled.

**X. References**


