Flashback-N Project
An extension to the FlashBack project

Qing Zhang
Summer 2004

Abstract

Flashback is a lightweight Operating System extension that allows a process to be rolled back and replayed. It is lightweight because it use a shadow processes to keep track of the in-memory state of a process. It also dynamically logs the interaction between a process and the system calls it makes. Using the in-memory state and the system-call -logs it enables a process to be deterministically replayed for software-debugging purposes without a high overhead. The work done on Flashback is implemented on the Linux OS. Flashback allows a user to create a checkpoint and rollback to that sole checkpoint and replay it deterministically. Flashback-N is an extension to the Flashback project by enabling users to make multiple checkpoints, and deterministically replay the process at any of the checkpoints made.

Motivation and Background

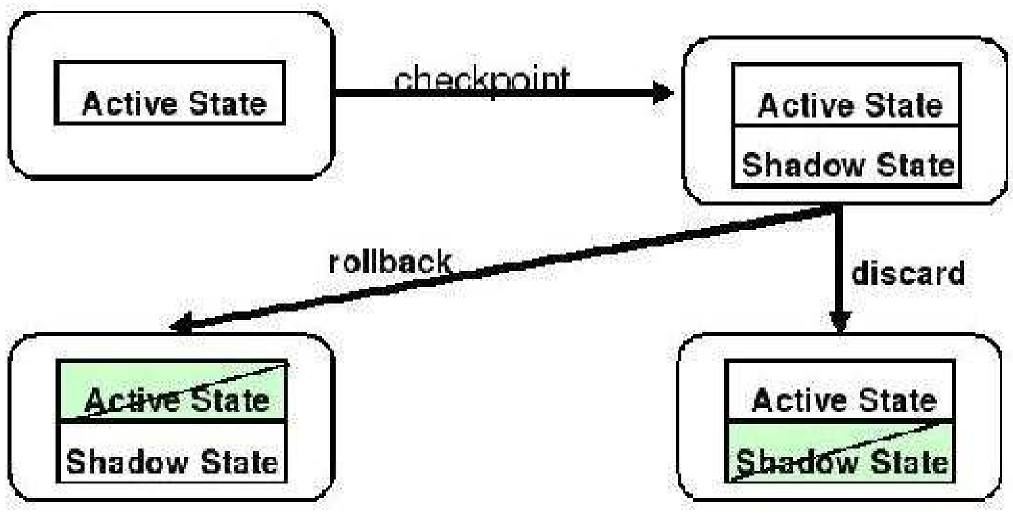
Reliability of software has always been a big issue. Software bugs are said to account for more than 40 % of system failures, and costs the U.S. economy $59.5 billion annually. Therefore it is a necessity to develop tools that will help detect and correct these problems. There exists debugging tools such as gdb, which will allow a user to trace the program. However, it is very hard to track bugs in faulty programs that ran for hours or even days at a time. Especially for bugs that are hard to reproduce; i.e. programs that require the specific hardware configurations and the exact user inputs to crash. These types of software bugs can be very frustrating and time consuming to find. This lead to the birth of Flashback. It's an low-overhead debugging tool that allows programmers to deterministically re-execute a process at a specific checkpoint. It is useful in that it allows a programmer to deterministically replay the faulty process. However, the current implementation of Flashback only allows one checkpoint per process. It is sometimes difficult to determine where the sole checkpoint should be placed in a program to efficiently find the bug. Also for programs that run for hours at at time it will be difficult to debug even if re-execution is enabled. Flashback-N extends the Flashback project by allowing multiple checkpoints to be made. The programmer can therefore efficiently rollback to any specified checkpoint given a checkpoint number.

Project Details

The three main functions provided by Flashback-N are checkpoint(), rollback(x), and replay(x). checkpoint() allows a user to capture the execution state at that checkpoint, and rollback(x) allows the program to rollback to any previous execution state specified by the checkpoint number x, also without calling replay(), a process can be executed at the checkpoint without deterministic replay. The replay(x) will perform a deterministic re-execution of the process at the x-th checkpoint.

Shadow Processes

The framework for keeping track of a process's execution state in memory for Flashback-N is using shadow processes. The shadow processes are created by the operating system. When a checkpoint is called a shadow process is created, which is an exact replica of the current running process. The shadow process is suspended and kept within the running process's task_struct. Multiple checkpoints are kept as an array. They act as snapshots of the running process at different stages i.e each checkpoint. Once rollback(x) is called, The program rolls back to the x-th checkpoint by using the x-th shadow process. When replay(x) is called, the current process is terminated and the selected shadow process will be activated. Any previous checkpoints will then be discarded. Also once a process is terminated all of it's shadow processes are deleted. The diagram illustrates the steps take for re-execution of a process for one shadow process.



Logging Mechanism

In addition to using the shadow processes to keeping track of the in-memory state, logging is also used to do successful deterministic replay. All the system calls made by the check pointed process are captured and recorded. Not only are the the system calls recorded the side-effects of the calls made by the running process are also kept, to enable deterministic replay. The main idea behind the logging is to load a kernel modules that act as a kernel handler wrapper which records the action of each system call made by a process. When replay is called the series of events are replayed by reading off of the kernel logs for a specific checkpoint. A process is logged at each checkpoint and when rollback and replay is called, the log files index at the correct checkpoint for deterministic replay at that checkpoint.

Automated Checkpoints

Flashback-N also implements automated checkpoints system to allow for timed checkpoints for large programs that run for a long period of time. The implementation is done GDB. It allows a user to checkpoint their program automatically and rollback to any checkpoint. However, more testing and fine tuning is needed for this feature. Please refer to the Future Work.

Changes to GDB

Flashback-N also made modifications to GDB. The three functions checkpoint(), rollback(x), and replay(x) are added to GDB. One can compile a program with GDB and break and dynamically insert multiple checkpoints to a program and deterministically replay it at any of the checkpoints by invoking the rollback, and replay function inside of GDB. This is a very helpful feature for debugging, because not only does the user get to use the powerful deterministic replay capabilities of Flashback-N he/she can use GDB functions to debug the replayed version of the process. For additional changes to GDB refer to the Further Work section.

Conclusion

Flashback-N is an extension to the Flashback project. It is implemented in the 2.4.22 version of the Linux kernel. It a lightweight Operating System extension software debugging tool for rollback and deterministic replay. It allows multiple checkpoints to take place for software debugging purposes. Flashback-N is also implemented in GDB.

It currently supports automated checkpoints though it's not tested thoroughly.

Further Work

Future work involves fine tuning of automated checkpoints by retrieving pertinent information about each checkpoint and displaying each line that the checkpoint is made in GDB. Also allowing variable time length of the checkpoints, and testing to see how many checkpoints should be kept per process before starting over, and perhaps algorithms to determine which checkpoint should be kept. Currently the max is 100.

Also, currently there are little done in Flashback-N to support networking capabilities, such as message passing. It would be nice if it could be run on network applications such as Apache to test for errors.

Acknowledgments

I would like to thank YY Zhou. She is my adviser for this project. She offered many helpful advice and encouragement on this project. I would also like to thank Sudarshan Srinivasan for all the help and tips he gave on this project, and other graduate students in the OPERA group. I would also like to thank Nancy Amato, DMP, and NFS for the support and funding which in turn gave me this awesome learning experience.

References

[1] Srinivasan, Kandula, Andrews, Zhou *Flashback:* A Lightweight Extension for Rollback and Deterministic Replay for Software Debugging.