Tolerating Hardware Device Failures in Software resume

Louisa Bessad

4 juin 2014

The problem, his importance, is it new ? Nowadays systems need to use drivers, those drivers permit access to devices. When a device fails, some drivers can crash or hang. In this way it was demonstrated that most of the systems failures are due to hardware devices failures. The major part of those device failures are transient. It was showned that drivers which tolerte device failures improve the reliability of the systems.

To control the hardware failures stress testing is not enough, we need fault-injection testing. Machines that can run these tests are limited because the presence of an instance of the device is needed.

Driver failures has been already studied, but we found only two approaches which does not check inputs from the driver :

— static bug finding which analyzes the interface between the driver and the kernel to avoid system crash.
— run-time fault tolerance

Most of the time these tools needed new operating sytem or new driver models. They also used memory detection to find hardware failure. Moreover they find the failure but do not fix them. When these tools works on reducing faults on the interface between driver and device, it was needed to have a specific interface and a specific language To insure inputs checking from the driver, have a failure detection before corruptions appears and fix device failure Carburizer was created.

The solution Carburizer is an automatic code manipulation-tool in association with runtime. Its goal is to improve the reliability of a system. So Carburizer takes a driver, lists bugs that can occur on this driver and suggests a driver without those bugs. Most of the time Carburizer is able to fix bugs that it had found. In this way it creates a recovery function to avoid the failure and inserts a calling to this function where the bug can occur. In this way it scans the code to :

1. locate code which depends on inputs from device by :
   — consulting table of functions which are know to make I/O and evaluating the return value

— searching variables (used as pointer or index of an array) and structures dependant an input from the device
— looking for loop waiting for a hardware device state without timeout. Carburizer considers as timeout a variable incremented or decremented and used as exit condition on the loop.
— searching panic calling and replace it by a recovery function

2. add code to validate the unchecked data received from device or timeout to avoid infinite loop. When the timeout expires a recovery function is called
3. verify device responsiveness by runtime service
4. report all device failure. When Carburizer is uneffective it allows correction by programmer. Carburizer is able to find already existing report systems.

Carburizer provides an automatic recovery system : the runtime service. The runtime service restores drivers and devices to functionning state by using agents kernel : shadow drivers. When a fuction is called between the driver and the kernel the shadow driver stores the state of the driver. The runtime service also allows to handle interrupts. By generating too many interrups devices can hang because when the handler interrupt is running there is no useful work. In an other side devices that does not generate interrupts can became instable or useless. Moreover the runtime service allows to find stuck interrupts. Thisissue does not lead the runtime service to restore the driver, it will only disable interrupt request line.

Results Carburizer found 992 bugs on Linux drivers probably caused by hardware failures. Over 992, 903 bugs are corrected by Carburizer by the insertion of code and runtime recovering but 58 are false positives. The others bugs involve intervention from a programmer because there are due to dynamic-array. The rate of false positive is 7.4%. About reporting failures, Carburizer finds 2383 locations where drivers detect failure by timeout or comparisons or range test. Only 781 errors are reported by drivers, Carburizer includes reporting code to the rest. Moreover the overhaed caused by using Carburizer is insignificant.

Point of view When Carburizer detects code depending on inputs from device it does not checked the content of the code. If a variable is used as an index array several times Carburizer checks the value of the pointer only the first use. If bounds are known it just checks that value of the pointer is included on it to fix possible bugs. Whereas with dynamic arrays it can only report the bug and tests if the value is NULL because bounds are unknown. It just modifies the code to avoid failure. Moreover when a variable depends on inputs from device there are notified as a possible bug, but if the variable is never used this notification is useful and create a false positive. Carburizer creates false positives and false negatives. A false positives occur when Carburizer does not detects validity check whereas the code contains one. It appends with infinite loop detection.

Carburize may not recognize a timeout inserted by the programmer if it is a non standard integer and Carburizer adds one. We can find false positives when the array has exaclty the same size that the index'range but it does not create a false failure detection by calling recovery function for isntance. There is also false positives when index array is calculate by shift operations. Most of the false positives which are found on dynamic arrays are due to calculate index by shifting or others operations which are not mask or comparaison. Repairs a false positives have no impacts on the execution code, it is just useless. Carburizer does not check when variables which depend on input from device are passed as an argument and not as return value. This is a false negatives.

Sometimes driver detects failures missed by Carburize, these are false positives too. This is due to the fact that :

— read operations can be wrapped in a single function which is not checked by Carburizer

— Carburizer cannot find checking system when the validation is implemented on a separerate function.

Few tests have been made with Carburizer, plateform are only network drivers or sound drivers and driver in the Linux 2.6.18.8 kernel distribution. Most plateform could be tested, moreover Carburizer is still not full automatic. It increases the number of failure logged which it was suggested to report device with bad functioning not only device with failure. By increasing number of failure logged Carburizer alloww to facilitate administrators' work. Carburizer suppose that when a hardware failure is detected by the driver, it will correctly respond to that failure. Carburizer does not handle crashes that occur when a device received a flag from the driver whereas it is not ready.