1st EuroSys Authoring Workshop, Leuven, Belgium, Apr. 2006.

REINCARNATION OF DEAD DEVICE DRIVERS IN A FAULT-TOLERANT OPERATING SYSTEM

Jorrit N. Herder
<jnherder@cs.vu.nl>

Commodity operating systems such as Windows fail to offer a dependable computing platform. Due to the lack of fault isolation in the kernel, a local failure can easily spread and take down the entire system. Studies on software reliability report fault densities of 6 to 16 bugs per 1000 lines of operating systems code, and device drivers---that comprise the majority of code---have a reported error rate that is even 3 to 7 times higher. These estimates translate to tens of thousands of potentially fatal bugs in the multimillion-line kernels of monolithic.

Therefore, we propose a new, strongly compartmentalized, multiserver operating system that is designed to be fault tolerant. We greatly reduced the size of our kernel to under 4000 lines of code by moving most of the code (and thus most of the bugs) to user space. Each server and each device driver runs as an unprivileged user-mode process and is encapsulated in a private address space that is protected by the MMU hardware, so that faults in one module cannot corrupt another.

Moreover, we have explicitly designed our system to recover from common failures in servers and drivers, transparent to the applications and without user intervention. All servers and drivers are guarded by a special server known as the reincarnation server. If a problem is detected, the reincarnation server looks up the associated policy and can restart the failing or failed component with a fresh copy. Since many I/O bugs tend to be transient, due to rare timing, deadlocks, and the like, in many cases restarting the driver will cure the problem. This is how fault tolerance works: a fault is detected, the faulty component is replaced, but the system continues running all the time.

In this paper, we will describe the recovery mechanisms that allow reincarnation of dead device drivers in more detail. Among other things we will discuss the set of faults we can deal with; the policy scripts that guide the recovery process at the reincarnation server; how failed components can recover their state after a restart; how dependant components, such as the file server and network server, handle driver failures; and what the application-level recovery scenarios for different kinds of device drivers look like.

We have not only designed, but also implemented the system and measured its performance. As an example, we used wget to retrieve a 512-MB file from the Internet and repeatedly killed the Ethernet driver to simulate crashes with intervals ranging from 1 to 15 sec. Due to our transparent recovery mechanisms, the transfer succeeded in all cases without user intervention. The mean recovery time was 0.36 sec. The loss in throughput due to driver failures ranges from 25% with one failure per second to just 1% in the best case.