www.cmpnet.com
The Technology Network

Electronics times

Cracking the code

By Chris Edwards, EE Times UK
Jun 14, 2002 (6:41 AM)
URL: http://www.electronicstimes.com/story/OEG20020614S0072

Testing code is increasingly an issue, but is it worth it? Software complexity is racing ahead of hardware, even following the breakneck pace of Moore's Law. Hardware design experts worry about an increasing gap between what can be achieved at the semiconductor process level and at the hardware design level. But in terms of lines of code, the amount of software going into an electronic system is increasing much faster than that of hardware. According to the MedeaU European electronic design automation roadmap productivity in software doubles every five years, but the lines of code doubles every year. With millions of lines of code being put into systems, there is a serious question over whether the system can be tested even close to exhaustively. And whether it is worth it.

For safety-critical systems, testing and formal verification techniques will be needed in concert because any software failure is unacceptable. But for many systems, it is possible to recover from a software crash with only minimal loss of data if you assume that most crashes are caused by an unusual combination of events and are not systemic. This is the theory that lies behind a new push in the science of high-availability systems where software, not hardware, is considered to be the main culprit behind a failure.

High availability has become a key part of the CompactPCI hardware environment based on the core bus as well as extensions such as the PICMG 2.16 switched Ethernet backplane. Hot-swap hardware and low-level software that sits on top of the host OS monitors hardware in the system to watch for a failure.

The most common technique is a heartbeat protocol that works out whether a card is still running. If the heartbeat stops, the card has probably died. The system then moves tasks to a spare and tries to recover as many channels as possible. Even if the host processor fails, it is possible to fail-over to a hot spare, as long as there is a common, replicated database with enough information to allow a restart.

High availability support tends to be platform-specific. So environments have been migrating to 'alien' OSs to try to get fail-over support for host processors and intelligent I/O cards. OSE Systems has been porting its implementation of the heartbeat protocol — the link handler — and its support software to host OSs such as Solaris. The idea is to get a common high availability environment that can watch for software and hardware problems.

Indes, OSE Systems' distributor in Benelux worked with a major telecoms supplier on combining OSE running on some 100 processors with a standard OS running the management part of the system.

Gerard Fianen, general manager of Indes, said: "The management part of the system is based on Solaris. That is quite typical for this kind of system. [Within the OSE portion], they extensively use link handlers. With them, they can extend the programming model to a multiprocessor platform that is transparent over many CPU configurations with supervision. The customer wanted to integrate the management channel, connected over Ethernet."

The link handler was ported to Solaris, with the OSE programming model forming part of the Solaris environment. The initial platforms for the OSE Gateway environment are expected to be Solaris, Windows 32 and Linux.

"But it can be expanded to VxWorks and pSOS," said Fianen. "OSE checks if the process on the other side still exists and informs the local application whether the remote application has vanished."

It is possible to give the Solaris side of the environment the ability to fail over to a working system if a piece of hardware fails. He added: "A specialised version uses dual Ethernet channels with one link handler. It is not a standard product but could be part of a services offering for other prospects."

To make high availability less platform-specific, a group of OS vendors, telcos, hardware suppliers and middleware writers clubbed together to form the Service Availability Forum. The group aims to define a set of software specifications on which dependable services can be built.

High availability considerations are expected to show up outside the telecoms and datacoms environments because of the increasing difficulty of testing software and deal with environments where software may be downloaded and run dynamically.

Inder Singh, president and CEO of LynuxWorks, said: "The ability to recover from software crashes will be seen in all products. Microsoft has sensitised a lot of people to software crashes. There are more problems now [in high-availability systems] from software crashes than hardware reliability."

Previously, the focus in high availability and fault-tolerant system design circles has been on the impact of hardware crashes. "We will probably end up offering that support in both our OSs: BlueCat Linux and LynxOS," said Singh.

Software protection in embedded systems started with the ability to defend the core OS from errant tasks. That thinking drove the development of memory-protected OSs, such as QNX's eponymous OS, Wind River's VxWorks AE and forms of embedded Linux. Like process model OSs run on workstations, these OSs' memory management units build firewalls between each task so that data cannot be corrupted by a different program.

The problem that has faced most operating systems that use an MMU is that any context switch incurs a high overhead because the references to the memory pages that a task can access need to be flushed and loaded for the new task. But this apparent overhead has gradually reduced, partly through a relentless rise in clock speed and partly because newer embedded processor cores, such as the ARM10, as well as those that are derived from workstation processors, have employed techniques to reduce the cache overhead on virtual memory.

Concerned about whether the overhead should be incurred on every task switch, Wind River decided to make it possible to customise the protection between them.

Instead of each process being given its own area of protected memory, the data and code segments can be separated case-by-case. The Tornado development environment is used by the developer to set up the protection levels.

Some OS implementations use unmapped memory traps to stop programs with badly assigned pointers running amok. Strictly speaking, a null pointer, a common error in C programming, is a pointer with an uninitialised value. It does not have to be zero, but many are. By refusing to map page zero, an OS can stop instances of null pointer abuse from trashing the interrupt vector table, which often sits in the zero page in physical memory.

In concert with a heartbeat protocol through which applications communicate with the kernel, these techniques can be deployed to build systems that, although not quite having the responsiveness of an N+1 high-availability system, can pick up a dead or wayward task and restart it more gracefully than forcing users to actively reset the OS.