

Real-Time Linux®: The RedHawk™ Approach

Jason Baietto
Concurrent Computer Corporation
2881 Gateway Drive, Pompano Beach, FL 33069

1. Introduction

Concurrent Computer Corporation has been selling real-time solutions in various forms for over 40 years. While the majority of Concurrent's past real-time solutions were proprietary, Concurrent fully embraced free and open source software in 2000 and shifted focus to a new real-time product based on GNU/Linux called RedHawk Linux. A key goal of RedHawk Linux is to provide the same level of real-time support that Concurrent's existing customer base depends upon, while leveraging open source software with minimal modifications in order to reduce development costs. The RedHawk Linux solution meets Concurrent customer requirements and allows for continuous Linux updates by providing features and software layers that isolate applications from the dramatic level of change occurring in the Linux community.

2. Overview

RedHawk Linux is not a general purpose Linux distribution. At the core of the product is a real-time enhanced Linux kernel that can in theory sit on top of any Linux distribution. The RedHawk kernel is based on pure stable kernel releases from kernel.org and is then enhanced by including various open source patches as well as locally developed features. Concurrent frequently re-bases RedHawk to the latest Linux kernel releases in order to provide community fixes and enhancements to customers.

RedHawk Linux also contains a set of user-level packages including libraries, headers, documentation and commands that provide access to the kernel's real-time features. The APIs provided are an evolution of the APIs that have been developed during Concurrent's long history of providing real-time solutions. These APIs provide real-time application developers with a feature rich, stable and time proven interface. The use of this interface has allowed many of Concurrent's customers to move their applications from Concurrent's older proprietary solutions to RedHawk, and then move again from a RedHawk based on the 2.4 Linux kernel series to a RedHawk based on the 2.6 Linux kernel series, with very minimal code changes.

Many of Concurrent's customers wish to develop and deploy real-time applications written in the Ada language, still widely used by the United States military. Concurrent's [MAXAda](#) environment provides a feature rich Ada development environment that takes full advantage of the real-time enhanced kernel included with RedHawk Linux. Concurrent also provides a high-performance FORTRAN development environment, along with the standard GNU C and GNU C++ development environments.

An optional tool suite called the NightStar™ Tools is also available and is a big part of the value-add of RedHawk. The [NightStar Tools](#) provide debugging and analysis tools that are specifically targeted to real-time application development and system tuning, again taking full advantage of the real-time

enhanced kernel included with RedHawk Linux. Note that the NightStar Tools can debug, analyze and tune applications written in C, C++, Ada and FORTRAN.

Concurrent also enhances RedHawk's value via hardware integration, custom engineering services, maintenance contracts and extensive end-user documentation.

3. Shielding

At the heart of Concurrent's approach to real-time is the concept of processor shielding, which is the reservation of specific CPUs in the system for real-time processes. For processor shielding to work there obviously must be more than one CPU in the system, and for nearly three decades Concurrent's hardware solutions have consisted of sophisticated SMP architectures often with NUMA support. Now that multi-core processors are widely available, the number of platforms that can exploit shielding has exploded, and this has enabled inexpensive single processor computers to become highly effective real-time solutions.

The RedHawk kernel combines existing shielding support in the Linux kernel with locally developed enhancements and provides convenient user-level tools to configure and control processor shielding. For example, the supplied *shield* command can be used to dynamically block processes, generic interrupts and even the local timer interrupt from occurring on a specified set of CPUs. Real-time processes running on a dedicated CPU can be ensured of the full bandwidth of the processor without interruption, and importantly, with virtually no cache interference, dramatically improving the process' determinism. No other real-time mechanism can offer this level of determinism.

It is important to realize that Concurrent's processor shielding implementation consists of a relatively small set of kernel extensions and does not involve a dramatic redesign of the entire kernel spin-lock and interrupt model (as is the case with the proposed PREEMPT_RT patch being developed by Ingo Molnar). This low-overhead approach dramatically enhances the viability of the solution and minimizes any impact to system stability and complexity, something that has not been the case up to this point with the PREEMPT_RT approach.

4. Preemption

RedHawk provides a user-space mechanism that allows a running process to tell the kernel that it cannot be preempted for a brief period of time. This feature is implemented by having the process register a special memory location with the kernel called a *rescheduling variable*. The process can then control preemption simply by writing to or clearing the rescheduling variable, without requiring any system calls to be performed. This feature is used extensively throughout the user-level libraries provided with RedHawk, and in particular it is used to provide a reliable and fast implementation of user-level spin locks.

5. Graphics

RedHawk includes graphics drivers that have been enhanced to minimize the need to cause cross-processor interrupts. Both open source drivers and proprietary drivers such as the popular drivers from nVidia have been enhanced and are included. The interrupt reduction has largely been achieved by providing a mechanism to preallocate graphics pages at boot time, so that floods of TLB flushes will not need to be performed as new pages are mapped by the graphics driver.

6. Synchronization

RedHawk provides a fast and efficient sleep/wake mechanism that can be used between a group of cooperating threads or processes. This service is based upon the open source Post-Wait patch developed by SGI, however it has been enhanced to take advantage of rescheduling variables to ensure atomic wakeups and improve determinism.

7. Schedulers

RedHawk provides the standard Linux process scheduler as well as a locally developed *frequency based scheduler* (FBS). The FBS is a high-resolution task scheduler that enables the user to run processes in cyclical execution patterns. The FBS controls the periodic execution of multiple, coordinated processes utilizing major and minor cycles with overrun detection. A *performance monitor* (PM) is also provided to view CPU utilization during each scheduled execution frame.

The FBS interface has been developed and enhanced over decades and now represents an ideal scheduler optimized for classic hard real-time applications such as data acquisition, simulation and process control. Applications written for the FBS have required little or no change over the last several years, whereas in that same time the Linux scheduler has seen many re-writes with subtle changes to both load balancing heuristics and scheduler fairness.

8. NUMA

RedHawk systems that use the AMD Opteron architecture can take full advantage of the architecture's NUMA facilities. Concurrent has enhanced the *run* command to allow the specification of memory policies along with other process environment settings, creating a single tool which can conveniently be used to bind a process to both a specific processor (or core) and a specific NUMA node. The next version of RedHawk will provide the ability to duplicate pages, ensuring that the full set of pages for a given process are local to a single NUMA node, further improving process determinism by eliminating all access to and from foreign nodes.

9. Hardware

Concurrent is committed to using *commercial off-the-shelf* (COTS) hardware in real-time solutions whenever possible. Unfortunately, a large percentage of COTS hardware is not suitable for real-time use for several reasons, including:

- The system BIOS does not allow fine-grained control over PCI slot IRQ assignment
- The system has devices which cannot be disabled yet are hard-wired to share IRQs with PCI slots
- The firmware generates periodic SMI interrupts which cannot be disabled
- The hardware lacks a mechanism to generate an NMI for debugging a hung system
- Some chipsets have inherent issues with real-time (e.g. unfair memory access across CPUs)

Systems qualified by Concurrent to be real-time compliant are called iHawk™ systems and are included in the list of systems approved for real-time solutions. Qualified systems also become integrated into Concurrent's *automated nightly test system* ([ANTS](#)). New versions of existing iHawk systems are

periodically re-qualified to ensure that subsequent board-level changes and new chipset revisions do not impact real-time performance.

The list of qualified iHawk COTS systems includes offerings from many vendors including Dell, HP, Supermicro, Tyan and Newisys. RedHawk is also available on various IBM BladeCenter blade configurations.

Many of Concurrent's customers also have a long history of using VME-based technologies, and thus RedHawk fully embraces the VME standard. Concurrent offers a PCI-to-VME chassis which can be used to host many different standard VME I/O boards, connected by fiber optic cables resulting in exceptional run-time performance. In addition, RedHawk has been ported to several different VME-based boards running multi-core Intel processors, and thus customers who wish to use VME technology can choose between two different approaches - native VME or a VME I/O bridge from a RedHawk PCI-based platform. RedHawk offers many real-time optimized drivers for industry-standard VME cards, including analog-to-digital, digital-to-analog, discrete I/O, MIL-STD-1553 (avionics) and CAN bus (automotive).

Almost all iHawk systems shipped by Concurrent contain a version of Concurrent's locally developed *Real-Time Clock & Interrupt Module* (RCIM) which is available in both PCI and PMC form factors. The RCIM provides connections for external device interrupts, programmable interrupt generators and real-time clock timers that can trigger interrupts. Multiple RCIMs can be cabled together to provide distributed interrupts allowing the RedHawk FBS to synchronize processes across multiple machines with very high accuracy. The RCIM also provides a mechanism to synchronize system clocks and it can optionally be fitted with a GPS receiver and an oven-controlled crystal oscillator for extremely high precision.

10. Development

In addition to providing unmatched real-time performance, RedHawk is specifically designed to simplify both real-time application and kernel driver development.

To facilitate real-time application development, the kernels included in RedHawk have been instrumented with a low-intrusion *kernel trace* mechanism. Using Concurrent's NightTrace™ tool, kernel traces can be combined with user-level application traces to provide a highly detailed picture of both kernel and application events displayed on the same time-line. This provides unprecedented visibility into how the real-time application and the kernel interact, and is a tremendous aid in design, tuning and debugging.

Note that Concurrent's kernel trace mechanism is implemented by inserting permanent trace point macros at specific key places inside the kernel code. These macros can be conditionally disabled, and RedHawk provides pre-built kernels with and without kernel trace. It is important to understand that the kernel trace mechanism is primarily intended to be used for the analysis of real-time application behavior; in particular, it is not designed for generic kernel debugging and is not intended to be an ad-hoc kernel debugging tool like Kprobes. Conversely, Kprobes would be a poor mechanism to use to implement kernel trace for many reasons:

- Kprobes is not designed to collect large volumes of trace data efficiently; it uses *int3* traps, duplicates large portions of stack and locks a global hash table to look up unique handler functions causing excessive SMP contention.
- Kprobes is inflexible; it has limitations regarding where probes can be placed, and is limited to accessing only data that exists in registers at the time of the probe.

- Kprobes is inconvenient, especially for end users; it requires the compilation and loading of a kernel module each time a configuration change is made.
- Kprobes requires a high level of maintenance; maintaining a set of accurate trace points across evolving kernel versions would be very difficult.

Thus, while Kprobes may excel as a generic ad-hoc kernel debugging tool, it is unsuitable for the basis of a mechanism primarily intended to gather exhaustive kernel event details for the purpose of illuminating real-time application behavior.

The NightStar tools also include the NightView™ real-time optimized application debugger and the NightTune™ real-time application tuning utility. The combination of NightTrace, NightView and NightTune results in an integrated suite of tools custom tailored for real-time application development and tuning.

To facilitate kernel driver development, RedHawk includes support for the KDB and KGDB kernel debuggers. In addition, emphasis is placed on tools to create and analyze crash dumps from customer sites, and RedHawk has supported LKCD and now the kexec/kdump method of producing crash files.

11. Community

Concurrent has been actively involved in the free software and open source communities for many years. Concurrent currently sponsors open source projects including *run*, *cpuid*, and *nuu*, and members of the Concurrent kernel development team post bug fixes and feature patches to the Linux Kernel Mailing List. Previous patch submissions have involved high-resolution timers, POSIX timers, kernel debuggers, RCU, POSIX message queues, graphics drivers, ptrace, NUMA and NFS and some of the patches have now been incorporated into the latest versions of the Linux kernel.

12. Future

Many more features and enhancements are planned for future versions of RedHawk, including real-time enhancements to Infiniband, enhanced kernel trace, enhanced NUMA support, support for using RedHawk in real-time clusters, support for user-level device drivers, the migration of GFS to GFS2, etc. with priority always given to those features that are most requested by Concurrent's real-time customer base.

Concurrent is committed to keeping close to the changes being accepted by kernel.org, and RedHawk will make full use of the PREEMPT_RT patch when it has proved itself to be stable and powerful enough to be accepted into the kernel.org mainstream kernel. A RedHawk kernel that combines the best of PREEMPT_RT with Concurrent's locally developed features will provide customers with a very powerful and versatile environment for developing deterministic and low-latency real-time applications.