A Concurrent Real-Time White Paper

2881 Gateway Drive
Pompano Beach, FL 33069
(954) 974-1700
www.concurrent-rt.com

Real-Time Linux: The RedHawk Approach

By: Jason Baietto

Chief Software Architect

December 2019

Abstract

Please read this white paper to learn about Concurrent Real-Time's revolutionary Linux RTOS that safeguards deterministic application behavior while minimizing impact on system stability and complexity. At the heart of Concurrent's approach to real-time Linux is the concept of processor shielding, a means to reserve CPUs for real-time processes. Contrasts to standard Linux and alternative scheduling strategies such as PREEMPT_RT are discussed.

Background

Concurrent Real-Time has delivered mission-critical, real-time solutions for over 50 years. While many of Concurrent's past real-time products were proprietary, Concurrent fully embraced open source software in the year 2000 and shifted focus to a new real-time product based on GNU/Linux called RedHawk Linux. A key goal of RedHawk Linux was to provide the same level of real-time support that Concurrent's customer base relied upon, while leveraging open source software with minimal modifications in order to stay as close as possible to the kernel.org source code. RedHawk Linux meets Concurrent customer requirements and allows for continuous Linux updates by providing features and software layers that isolate applications from the on-going levels of change occurring in the Linux community.

Overview

RedHawk is a high-performance Linux distribution for x86 and ARM64 platforms. At the heart of the product is a real-time enhanced Linux kernel that features popular Linux user environments, i.e. RHEL, CentOS, Ubuntu, Debian. The RedHawk kernel is based on long-term support versions of kernel.org and is then enhanced by including various open source patches as well as Concurrent-developed features. Concurrent regularly re-bases RedHawk to the latest Linux kernel releases in order to provide community fixes and security enhancements to customers. Concurrent closely monitors the versions of the various Linux distributions, ensuring that support for both the kernel and Linux distributions is as up-to-date as possible.

RedHawk Linux 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 were 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 user 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 also move from early RedHawk versions to the latest versions with very minimal code changes. In addition, Concurrent offers support services to help migrate users of other OS platforms to RedHawk Linux.

RedHawk Linux supports an optional package called the NightStar, a powerful, integrated tool suite for developing time-critical applications. NightStar tools run with minimal intrusion, thus preserving application execution behavior and determinism. Users can quickly and easily debug, monitor, schedule, analyze and tune applications in real-time. NightStar GUI-based tools reduce test time, increase productivity and lower development costs. Time-critical applications require debugging tools that can handle the complexities of multiple processors and cores, multitask interaction and multi-threading. NightStar's advanced features enable system builders to solve difficult problems quickly. NightStar tools can debug, analyze and tune applications written in C, C++, Ada and FORTRAN.

Concurrent also offers fully-integrated iHawk real-time computer systems running RedHawk Linux, custom engineering services, maintenance and extensive end-user documentation. RedHawk training at all levels, from introductory Linux programming to advanced real-time techniques, is available at Concurrent's home office or at a customer site.

Processor Shielding

At the heart of Concurrent's approach to real-time is the concept of processor shielding -- the reservation of specific CPU cores in a system for real-time processes. The RedHawk kernel combines existing shielding support in the Linux kernel with Concurrent-developed enhancements and 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 interrupts 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, thus dramatically improving process determinism. No other real-time mechanism can offer this level of guaranteed low latency.

It is important 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 PREEMPT_RT patch). This low-overhead approach dramatically enhances the viability of the solution and minimizes any impact to system stability and complexity.

Preemption Reduction

RedHawk provides a user-space mechanism that allows a running process to tell the kernel that a process cannot be preempted for a brief period of time. This feature is implemented by allowing the process to register a special memory location within the kernel called a rescheduling variable. The process can then control its preemption simply by writing to or clearing the rescheduling variable without the need for any system calls. 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.

Interrupt Reduction

RedHawk Linux has been extensively tuned to minimize the number of interrupts that occur on a CPU during the execution of a real-time process. This tuning has been implemented as described in the following subsections.

Interrupt Blocking

Real-time applications running under RedHawk have the ability to block delivery of all interrupts to the CPU that an application/thread is running on for short periods of time. In standard Linux, this ability to block interrupts is available only to kernel code, such as device drivers. This blocking feature is useful for those applications that have short code sequences with short run times that have very tight timing requirements.

TLB Flush Reduction

The *vmalloc* kernel virtual address space is used for a variety of purposes, such as temporary buffer space, kernel data structures, and *ioremap* operations. When any previously allocated *vmalloc* virtual space area is freed, a cross processor interrupt (CPI) is issued to all the other CPUs in the system so that each CPU will issue a translation look-aside buffer (TLB) flush instruction to remove any cached translations to the previously used *vmalloc* virtual area.

While these types of TLB flush CPIs typically take only a short period of time to execute, a series of multiple back-to-back TLB flush CPIs can occur and thus greatly lengthen the total amount of time that a CPU is processing TLB flush activity. This flurry of TLB flush CPIs has been observed to produce a noticeable lack of determinism on a shielded CPU.

The RedHawk *vmalloc* support code has been rewritten to provide a configurable, alternative *vmalloc* algorithm that greatly reduces the number of *vmalloc*-related TLB flush CPIs, thus eliminating this source of potential jitter on shielded CPUs.

The *vmalloc* support has been modified so that when new *vmalloc* space is allocated, a kernel virtual address that is higher than the last *vmalloc* allocation is selected whenever possible. Only when the search for free space wraps around to the beginning of the *vmalloc* virtual space area are TLB flush CPIs issued to remove any previously used translations. This ensures the impact to determinism is small and controlled.

Memory Pre-allocation

RedHawk includes NVIDIA graphics drivers that have been enhanced to minimize the need to cause CPIs. Interrupt reduction has largely been achieved by providing a mechanism to pre-allocate 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.

Latency Minimization

A fast, timely reaction to external events is an important feature of a real-time system. A system which exhibits this property is said to be a low-latency system. Areas that can cause poor latency are:

- Processor frequency scaling damages latency because it takes time for the OS to detect the need to increase the frequency of a slowed-down CPU. The OS then must step up the frequency in the proper pattern required, up to the normal high frequency of operation. During this step-up time, the CPU is not running at full capacity and hence is responding slower to an external event than it would have if it had already been running at full speed.

- Some multi-core processors have an additional power saving state called the C1 *extended* (C1E) state. This state is entered when all the cores are in the halt instruction. In this state, everything on the processor turns off, even the ability of the cores on that processor to generate a local timer interrupt. If the power saving state is allowed to happen, the OS must turn off high-precision POSIX timers in favor of the Hz-based low-precision POSIX timer system, since high-precision timers require the use of the (now disabled) local timer interrupt for their proper operation.

In order to preserve the desired low-latency characteristics of the system, RedHawk has code that prevents it from changing CPU clock frequency and from entering sleep states. In order to ensure that the local timer interrupt is reliable, RedHawk ensures it never allows the C1E state to be entered.

Eliminating the effects of power management has the additional benefit of ensuring a highly reliable time stamp counter (TSC) CPU register which is essential for real-time systems. This is a feature not provided by standard Linux.

Time Source Improvements

Real-time applications generally need a time-source that is:

1. Extremely fast to access
2. Resolution of at least one nanosecond
3. Accurate to within a few hundred nanoseconds of the correct time

Standard Linux does not provide a time source which meets all three of the above criteria. The closest provided is the standard clock-source based on the CPU's TSC register. The TSC is fast to access and also returns a precise number, but it too easily drifts away from the correct time, thus losing accuracy.

Alternatively, a high-quality externally-supplied time source, for example a GPS module attached to the PCIe I/O bus, may be extremely accurate internally but suffers from slow and varying access times due to the relatively long distance between devices on the PCIe bus and the CPU. This slow and varying access is a function of the varying traffic load on the PCIe bus and the memory bus to which the PCIe bus is attached.

RedHawk Linux solves these problems by implementing the concept of stacked clock sources. Applications still use the clock_gettime(2) system service to fetch timestamps, however, instead of internally having only a single clock source driving the service, there is a pair of attached clock sources. The primary clock source of the pair needs to be fast to access and needs to return a high-precision value. The TSC has these attributes so it is universally selected as the primary clock-source.

The secondary clock source needs to have the attribute of being extremely precise and extremely accurate, but it need not have the attribute of fast and unvarying in access times. For RedHawk, the ideal secondary clock source is Concurrent's Real-Time Clock & Interrupt Module (RCIM) PCIe card. More information on this card is provided in the Hardware Certification section below.

The primary clock source is always used by clock_gettime(2) to return the system time. RedHawk uses the secondary clock source to periodically recalibrate the primary. This recalibration keeps the primary from drifting too far away from the correct time. The recalibration algorithm takes into account the length of time it takes to read the secondary clock-source and has a compensation mechanism that minimizes the effects of the variation in read time.

Process Synchronization

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

Additional Schedulers

RedHawk provides the standard Linux process scheduler as well as a Concurrent-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 provided to view CPU utilization during each scheduled execution frame.

The FBS interface has been developed and enhanced over time 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 years, whereas the Linux scheduler has undergone many updates with subtle changes to both load balancing heuristics and scheduler fairness.

NUMA Optimization

RedHawk's NUMA optimization features can dramatically improve the determinism of real-time process memory access on modern NUMA architectures. RedHawk can automatically duplicate libraries and other modules as needed and hold them simultaneously in multiple nodes to maximize performance.

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 core and a specific NUMA node.

Concurrent has enhanced the *shmconfig* command to allow the specification of memory policies that control the NUMA node location of shared memory areas. The entire shared memory area may be assigned the same memory policy, or differing memory policies may be assigned to various address ranges, thus placing specific pages of the shared memory area into the NUMA node memory where it is most frequently accessed.

Concurrent provides the ability to memory shield one or more NUMA nodes with the *shield* command. When a NUMA node becomes memory shielded, the kernel will automatically migrate user pages between nodes so that only user pages belonging to processes that are

biased to execute on CPUs in the memory shielded NUMA node will be contained in that node's memory. Thus, process determinism is improved by eliminating remote memory accesses from processes executing on the memory shielded NUMA node and by eliminating remote memory accesses into the memory shielded NUMA node by processes executing on non-shielded NUMA nodes.

When a non-memory shielded NUMA node's memory becomes full, page allocation requests may overflow into another NUMA node's memory. In this case, a memory shielded NUMA node's memory could end up containing pages belonging to non-memory shielded tasks. To prevent this occurrence, when a NUMA node becomes memory shielded, the kernel automatically reorders the node zone lists so that page overflow allocations will be first satisfied using pages from non-memory shielded nodes. Memory-shielded node memory is used only as a last resort when all non-memory shielded nodes are low on memory.

Memory shielding may be combined with process, interrupt and local timer shielding to create a totally isolated NUMA node dedicated to real-time processing.

As an aid for application development, the RedHawk *run* command may be used to view the number of currently-mapped user pages in each NUMA node on a per-process basis for all user-space processes in the system or for a specific set of processes. Additionally, the *numapgs* command provides the NUMA node location of each page currently mapped into the user address space of a single process, as well as flags indicating whether each page is replicated or user-locked (*mlocked*).

Real-Time Networking

RedHawk's real-time networking feature can be used to assign real-time processes their own fully-dedicated and private TCP/IP stacks. Packets transmitted-to and received-from private TCP/IP stacks are completely isolated from all other networking activity on the system, ensuring that packets for real-time processes will move through the system with real-time priority, low latency and high determinism.

Private TCP/IP stacks require private transmit/receive ring buffers for each connection along with special Ethernet hardware that can quickly process and route packets into each connection's corresponding private ring buffers. Ethernet cards produced by Solarflare Corporation feature efficient packet routing into private ring buffers and RedHawk includes kernel-level and user-level support for fully utilizing these cards.

RedHawk Architect

RedHawk Architect is a powerful tool with an easy-to-use GUI that lets a developer choose the Linux and application modules to be included in RedHawk target images. Designed especially for embedded applications, users can select as few or as many packages as desired from many

different package groups. Architect allows the file system to customized and minimized for diskless operation using flash size under 1 GB.

Architect's cluster manager software allows users to install and configure systems as highly-integrated, high-performance computing clusters. Architect includes mechanisms for network PXE installing and for network PXE diskless booting of multiple nodes with the same version of RedHawk. RAMDISK-based diskless booting of nodes is also supported.

Architect creates and processes a configuration file defined by the user to perform actual RPM package installation. The tool prompts the user to insert the required RedHawk, NightStar, and Linux distribution media depending upon the features selected. RedHawk Architect will allow customization of the RedHawk kernel itself and provides a flashing tool for burning RedHawk and the user's application image onto a CPU board's non-volatile memory, DVD or USB flash. Architect can also build virtual target images for use with QEMU/KVM. This allows embedded images to be tested without a physical target system.

Real-Time Virtual Guests

RedHawk offers a virtual real-time environment called KVM-RT based on the Kernel Virtual Machine (KVM) hypervisor with RedHawk as the host operating system. A KVM-RT installation provides guaranteed real-time performance in virtual machines. Multiple RedHawk real-time guests and multiple Windows and non-real-time Linux guests are supported. KVM-RT can achieve worst case response times of less than 10 microseconds on certified hardware platforms. KVM-RT allows real-time multiprocessing applications to better utilize available computing resources.

KVM is a Type 1 hypervisor that is included with RedHawk Linux. RedHawk KVM-RT leverages the unique features of the RedHawk host for memory management, process scheduling, device access and I/O. KVM-RT creates individual threads for each virtual CPU allocated to a virtual machine. Real-time performance is guaranteed through RedHawk's shielding and scheduling mechanisms. I/O devices can be completely passed through to specific guests or para-virtualized to reduce the idle time of devices. Para-virtualized devices reduce the amount of necessary hardware without compromising throughput.

KVM-RT configuration tools are available to assist the user in allocating a platform's processor cores, memory, peripherals and other hardware resources to each virtual machine.

Hardware Support and Certification

Concurrent is committed to supporting commercial off-the-shelf (COTS) hardware in real-time solutions wherever possible. Some COTS hardware is not suitable for real-time use for a variety reasons, including the following:

- The system BIOS does not allow fine-grained control over PCIe slot IRQ assignment.
- The system has devices which cannot be disabled yet are hardwired to share IRQs with PCIe 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 chip-sets have inherent real-time issues (e.g. unfair memory access across CPUs).

Systems qualified by Concurrent to be real-time compliant are Concurrent iHawk™ real-time multiprocessors and Concurrent ImaGen™ real-time visual servers. These platforms are on the list of systems approved for real-time solutions. Qualified systems are integrated and tested by Concurrent's Automated Nightly Test System (ANTS). New versions of iHawk systems are periodically re-qualified to ensure that subsequent board-level changes and new chip-set revisions do not impact real-time performance.

The list of qualified iHawk and ImaGen systems includes offerings from many vendors including Dell, Supermicro, Tyan, Trenton and other COTS motherboard suppliers. RedHawk also offers real-time optimized drivers for many industry-standard I/O cards including analog input, analog output, pulse width modulation, digital I/O, MIL-STD-1553, ARINC 419, reflective memory, serial I/O and others.

Real-Time Clock & Interrupt Module

Concurrent iHawk real-time multiprocessors include the Real-Time Clock & Interrupt Module (RCIM), a multifunction card designed for time-critical applications that require rapid response to external events. Eight programmable timers and twelve input and output external interrupt lines are available. Any interrupt source can be distributed to other iHawks for synchronizing multi-system applications. The RCIM includes a high-resolution synchronized clock to provide a common time base across multiple systems. On-the-wire time stamps allow RedHawk to provide for high-resolution NTP synchronization. RCIM options include a GPS module for synchronizing with GPS standard time and high-stability crystal oscillators to provide for accurate timekeeping without an external time source.

Development Environment

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

Concurrent's NightStar is a powerful, integrated tool set for developing time-critical CPU and GPU applications. NightStar tools run with minimal intrusion, thus preserving application execution behavior and determinism. Users can quickly and easily debug, monitor, schedule, analyze and tune applications in real-time. NightStar GUI-based tools reduce test time, increase productivity and lower development costs. Time-critical applications require debugging tools that can handle the complexities of multiple processors and cores, multitask interaction and multi-threading. NightStar's advanced features enable system builders to solve difficult problems quickly. The NightStar tools include the NightView™ real-time optimized application debugger, NightTrace event analyzer, NightTune™ real-time application tuning utility, NightProbe data recorder, and NightSim application scheduler.

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

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. RedHawk also 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 is 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 as a mechanism primarily intended to gather exhaustive kernel event details for the purpose of illuminating real-time application behavior.

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. RedHawk supports LKCD and the kexec/kdump method of producing crash files.

RedHawk makes the standard user-level device driver (UIO) layer available, complete with documented working examples of user-level driver code, to further simplify kernel driver development.

Community Involvement

Concurrent has been actively involved in the free software and open source communities for many years. Concurrent currently sponsors open source projects including *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 official Linux kernel.

Conclusion

More real-time features and enhancements are planned for future versions of RedHawk with priority given to those features most requested by Concurrent's real-time customer base. Concurrent Real-Time will continue to combine kernel.org releases with Concurrent-developed features to provide customers with a very powerful and versatile environment for developing deterministic and low-latency real-time applications.

About Concurrent Real-Time

Concurrent Real-Time is the industry’s foremost provider of high-performance real-time Linux® computer systems, solutions and software for commercial and government markets. The Company focuses on hardware-in-the-loop and man-in-the-loop simulation, data acquisition, and industrial systems, and serves industries that include aerospace and defense, automotive, energy and financial. Concurrent Real-Time is located in Pompano Beach, Florida with offices through North America, Europe and Asia.

Concurrent Real-Time's RedHawk Linux is an industry standard, real-time version of the open source Linux operating system for Intel x86 and ARM64 platforms. RedHawk Linux provides the guaranteed performance needed in time-critical and hard real-time environments. RedHawk is optimized for a broad range of server and embedded applications such as modeling, simulation, data acquisition, process control and imaging systems.

For more information, please visit Concurrent Real-Time at www.concurrent-rt.com.