**Technical White Paper**
DATA CENTER

www.novell.com

# Concurrent Real-Time Extensions

Powered by SUSE® Linux

concurrent

Novell®

**Table of Contents:**

# Rely on Real-time Capabilities for Linux*

**By choosing Linux and Concurrent RTE for your real-time data center, you can configure a modular, rich system that helps you consolidate servers, reduce costs and consistently achieve high quality of service.**

Many of today's businesses—including financial institutions, manufacturers, government agencies, telecommunications carriers and medical providers—require deterministic, real-time computing. In other words, their high-priority transactions, line processes, test data acquisitions and product simulations must execute accurately and predictably every time—at sub-second speeds. Because of these exacting requirements and the need for tight security and interoperability, a growing number of businesses are turning to Linux*.

You've witnessed firsthand the remarkable expansion and adoption of the Linux operating system. It's gained a significant footprint in the enterprise data center and currently supports thousands of applications. Now, Novell®, Inc. and Concurrent Computer Corporation have enhanced Linux to deliver the real-time capabilities you need. Concurrent Real-Time Extensions Powered by SUSE® Linux (hereafter called Concurrent RTE) is an industry-standard, real-time version of Linux for Intel- and AMD-based multiprocessors. This platform, an enriched distribution of the SUSE Linux kernel from Novell, provides you with guaranteed performance in time-critical environments.

Key features—including CPU shielding, kernel preemption, low latency and priority inheritance—help you achieve sustainable real-time performance on Linux. You can also rely on other important capabilities that improve quality of service and provide a solid foundation for a Service-oriented Infrastructure. These features include distributed shared memory, Xen virtualization and real-time cluster file systems, all of which are enhanced significantly by the low latency and deterministic processing in Concurrent RTE. By choosing Linux and Concurrent RTE for your real-time data center, you can configure a modular, rich system that helps you consolidate servers, reduce costs and consistently achieve high quality of service.

## Explore New Real-time Services

With recent real-time enhancements to the SUSE Linux kernel, Concurrent RTE showcases a highly flexible foundation. This foundation is essential for applications that require deterministic behavior. For example, financial-trading applications must respond to real-time events, such as new pricing, without any delays or interruptions. Likewise, in telecommunications applications, external signals must be processed instantly and without dropping a bit. It's also extremely important for government mission-critical environments such as missile defense, air traffic control, and emergency response systems to process their respective transactions without delay.

Now you can take a closer look at our real-time offerings for Linux. We outline the specific technical capabilities of Concurrent RTE, including enterprise services offerings. You will also see how Concurrent RTE delivers a solid foundation for your Service-oriented Infrastructure (SOI). You can use it to build the next generation of data-center computing, creating an on-demand infrastructure that is modular, highly responsive and easy to deploy and control.

## Concurrent Real-Time Extensions Features

Built on the powerful SUSE Linux kernel, Concurrent RTE delivers real-time performance and capabilities. These enhancements, while easy to deploy, should be configured by a professional who fully understands the

application requirements and can tune your system for optimal performance. Concurrent RTE features centralized management and the NightStar™ application development tools. The five NightStar tools in Concurrent RTE help you non-intrusively debug, monitor, analyze, tune and schedule time-critical applications.

Although extremely valuable for time-critical environments, Concurrent RTE is not appropriate for all application deployments. Real-time systems deliver extraordinarily predictable application response at the potential cost of overall system throughput. For example, in order to guarantee predictable response time for high-priority applications, normal system processing (e.g., daemons, interrupts and so forth) is occasionally delayed so the system can first complete these high-priority processes.

The following sub-sections outline specific modifications to Linux that allow highly predictable operating system behavior:

### *CPU Shielding*

Using Concurrent RTE, you can shield selected CPUs from unpredictable processing elements—such as interrupts and system daemons—that sometimes delay the execution of high-priority applications. For example, suppose a financial trading application was context-switched during a critical transaction. Without CPU shielding, this type of switch would delay transaction completion until the system could finish running the application.

By marking a CPU as shielded, you modify the affinity masks of all processes and interrupts in the system. In fact, the shielded CPU is removed from the affinity masks for most processes and interrupts. Only processes and interrupts that have explicitly set their CPU masks to run on the shielded CPU will be able to do so.

## Built on the powerful SUSE Linux kernel, Concurrent RTE delivers real-time performance and capabilities.

A shielded CPU provides a deterministic environment, one with predictable guaranteed responses to interrupts, timers, messages and network packets. Shielded CPUs are valuable because they simultaneously provide a standard Linux execution environment and superior quality of service.

When you run an application on a shielded CPU, you can be assured that regular system processing (such as clock/device interrupts or system daemons) will not negatively affect or interrupt application execution.

### *Kernel Preemption*

You can configure Concurrent RTE so that a high-priority process preempts a lower priority process already executing inside the kernel. Although standard Linux offers a configuration parameter for this behavior, most Linux distributions do not ship with the parameter enabled. With kernel preemption disabled, a lower-priority process executing inside the kernel would continue running until it exited the kernel. At that time, the high-priority process would begin running. By enabling kernel preemption, you'll see substantially faster response times for high-priority processes. When you have a real-time process ready for execution (perhaps in response to an asynchronous system event) the system will immediately interrupt or context-switch any lower-priority processes and start running the higher-priority real-time event.

### *Low-latency Enhancements*

To protect the shared data structures it uses, the Linux kernel relies on spin locks and semaphores to secure the code paths leading to those structures. To use spin locks,

**You can use Concurrent RTE to build the next generation of data-center computing, creating an on-demand infrastructure that is modular, highly responsive and easy to deploy and control.**

Concurrent RTE provides kernel tunables that allow a system administrator to set the priority of the kernel daemons that handle deferred interrupt processing.

**The five NightStar tools in Concurrent RTE help you non-intrusively debug, monitor, analyze, tune and schedule time-critical applications.**

the kernel must disable preemption—and sometimes interrupts—to prevent the deadlocks that would occur when an interrupt to the current code path also attempts to lock the spin lock. Disabling preemption negatively affects latency for high-priority threads of execution.

Concurrent has developed a standard methodology to identify the worst-case "preemption disabled" times. The low-latency enhancements applied to Concurrent RTE modify the algorithms in the identified worst-case preemption-disabled scenarios to provide better response times.

This capability directly impacts process-dispatch latency times for real-time applications. By refining kernel algorithms to allow efficient process-dispatch latency, a high-priority application can receive near instantaneous attention from the running system.

### *Priority Inheritance*

When used as sleepy-wait mutual-exclusion mechanisms, semaphores can introduce the problem of priority inversion. Priority inversion occurs when a high-priority process attempts to access a semaphore held by a low-priority process. If a medium-priority process preempts the low-priority process while it is holding the semaphore, the low-priority process will not be able to complete the critical section or unlock the semaphore. In this scenario, the low-priority process blocks the high-priority process, creating a "priority inversion" that can last for an indeterminate time (typically until the medium-priority process is completed).

To correct this problem, priority inheritance is implemented within the Linux semaphore semantics. Priority inheritance involves temporarily raising the priority of a low-priority process that is holding a semaphore when a higher priority process attempts to gain access to that semaphore. This ensures that the processes executing in a critical section have sufficient priority to continue execution until they leave the critical section.

### *SoftIRQ Enhancements*

Linux supports several mechanisms that are used by interrupt routines in order to defer processing that would otherwise have been done at interrupt level. The processing required to handle a device interrupt is split into two parts. The first part executes at interrupt level and handles only the most critical aspects of interrupt-completion processing.

The second part is deferred to run at program level. Interrupt-level processing is always performed at a priority greater than any program-level processing. The system can achieve better program scheduling latency by removing non-critical processing from interrupt level. The second part of an interrupt routine can be handled by kernel daemons, depending on which deferred interrupt technique is used by the device driver. Concurrent RTE provides kernel tunables that allow a system administrator to set the priority of the kernel daemons that handle deferred interrupt processing. When a real-time task executes on a CPU that is handling deferred interrupts via these daemons, it is possible to set the priority of the deferred interrupt kernel daemon so that a high-priority user process has a more favorable priority than the deferred interrupt kernel daemon. This delivers a more deterministic response time for the real-time processes that execute on this CPU.

## Programmatic Features

There are several additional kernel capabilities that ship in Concurrent RTE. You will receive supplemental libraries—and in some cases additional commands—to help you access these features:

### *High-resolution POSIX Clocks and Timers*

Concurrent RTE includes support for high-resolution POSIX clocks and timers. You can use POSIX clocks to perform time stamping or to measure the length of code segments. POSIX timers will allow your applications to schedule events on an individual or periodic basis. All Concurrent RTE timers are based on a high-resolution clock, and you can choose to specify times in relative or absolute time. In addition, Concurrent RTE provides high-resolution sleep mechanisms that you can use to put a process to sleep for a very short time.

### *Post/Wait (Fast Wakeup Services)*

In Concurrent RTE, you can use the post/wait service to quickly activate and deploy a blocked or suspended process. Without this service, extensive system-level processing can significantly delay activation (or "wakeup").

### *User-level Preemption Control*

When an application has multiple processes that run on multiple CPUs and rely on shared data, you need a way to protect that data against simultaneous access by more than one process. The most efficient mechanism for protecting shared data is a spin lock; however, your application cannot effectively use a spin lock if there's a chance it could be preempted while holding the lock. (This is because the preemptive process might also try to lock the spin lock, thereby causing a deadlock.)

To use spin locks efficiently, you can leverage a mechanism that quickly disables preemption. This mechanism specifies a location within a process's address space monitored by the kernel. When this address location is set to a non-zero value, the process will not be preempted in its execution.

When you combine spin locks with this preemption-disabling mechanism, you can effectively lock a critical section in a user application with just two memory accesses.

### *High-resolution Process Accounting*

The standard Linux kernel calculates a process's CPU-execution times with a very coarse-grained mechanism. This means the amount of CPU time charged to a particular process can be very inaccurate. The high-resolution process-accounting facility in Concurrent RTE provides a mechanism for accurate CPU-execution time accounting. It also delivers better application-performance monitoring. Standard Linux CPU accounting services make use of this facility in reporting CPU usage.

### *Support for Inter-system Time Synchronization*

Using Concurrent RTE, you can synchronize your Linux system time to GPS standard time. This is possible because Concurrent RTE supports the Concurrent Real-time Clock & Interrupt Module (RCIM) PCI card. The RCIM is a multi-function PCI card that offers two optional features for maintaining highly accurate system time. The first option is a GPS feature that keeps system time in synch with GPS standard time. The second option is a highly accurate crystal that enables you to synch system time across multiple nodes in a cluster or across multiple systems.

### *Privileged Operations*

The Pluggable Authentication Module (PAM) in Concurrent RTE provides a mechanism you can use to assign privileges to users

**Concurrent RTE provides a mechanism for accurate CPU-execution time accounting. It also delivers better application-performance monitoring.**

**Because Concurrent RTE is fully compatible with standard Linux user-level APIs, you can run applications written for other Linux distributions without recompiling or modifying them.**

and set authentication policies without having to recompile authentication programs. Using this feature, you can give a non-root user the ability to run applications that would normally require root-level privileges.

Some privileged operations—like the ability to lock pages in memory—are crucial for real-time applications. (This is because unanticipated page faults can significantly extend the time it takes to execute a code path.) However, the ability to lock pages in memory is a privileged operation in standard Linux, an action only permitted to the root user.

You can control access to page locking and many other privileged operations through a set of predefined rights. You simply assign these rights to individual users or groups, and privileges are granted through roles you define in a configuration file. A role is a set of valid Linux capabilities. You can use defined roles as building blocks for subsequent roles, with the new role inheriting the capabilities of the previously defined role. As you assign roles to users and groups, you can define the specific capabilities permitted to non-root users.

### *NightStar Application Development and System Tuning Tools*

The five NightStar tools in Concurrent RTE help you non-intrusively debug, monitor, analyze and tune time-critical applications. The non-intrusive nature of these tools is essential because you don't want the debugging or tuning processes to change your application's behavior.

With the NightTrace tool, you can review a time-based graphical display that represents both kernel- and user-level execution that occurred while the application was running. The NightView™ tool is a source-level multi-process and multi-thread debugger that supports features like low overhead conditional breakpoints and application patching while debugging. The NightProbe™ tool helps you easily monitor the values of variables in an executing application, the NightTune™ tool provides a graphical interface for kernel statistics and other system-performance metrics, and the NightSim™ tool helps you schedule and monitor time-critical applications that require predictable, cyclic process execution. In addition to symmetric multi-processors, NightSim supports multiple systems connected via Concurrent's RCIM. NightSim simplifies the creation of distributed scheduling and provides a single point of control for individual schedulers distributed across multiple target systems.

## System Deployment and Management

It's easy to deploy Concurrent RTE. You simply open the SUSE Linux management interface—known as YaST—and select the real-time kernel. This installs the real-time extensions on your SUSE Linux kernel. And because Concurrent RTE is fully compatible with standard Linux user-level APIs, you can run applications written for other Linux distributions without recompiling or modifying them.

YaST consolidates management, giving you a centralized way to activate clustering and real-time features. In addition, management tools from various original equipment manufacturers (OEMs) and independent software vendors (ISVs) easily interface with YaST. With this extraordinary interoperability and complete manageability, you can use the same operating system (OS) for everything from standard computing grids to high-availability clusters requiring sub-second failover capabilities. Linux already spans an enormous range of processing, including telephone handsets, desktops, servers, mainframes and much more. It's not surprising that it's the platform of choice for grids with thousands of processors.

## Building Your Service-oriented Infrastructure (SOI)

Novell and Concurrent are laying the groundwork for a Service-oriented Infrastructure (SOI), based on a highly deterministic computing environment. The critical features include CPU shielding, kernel preemption, low latency and priority inheritance, all described in this paper. Our SOI vision extends the quality of service (QoS) features currently available at a system level to a distributed computing environment. A number of other important services, including Distributed Shared Memory, Xen virtualization and real-time cluster file systems are enhanced significantly by low latency and deterministic processing.

To address data growth and server consolidation challenges, you need a solution that solves tomorrow's data center problems today. One such solution is a Service-oriented Infrastructure. It requires a strong modular foundation and enables you to add or subtract discrete components without affecting application performance. This flexible "building block" approach has tremendous benefits from a cost perspective. High demand will ensure that these building blocks are primarily, if not exclusively, comprised of commodity components. This will allow OEMs and ISVs to design with an out-of-the-box mentality. Likewise, you will be free to pick only the components you need as you custom design your own system.

With its real-time capabilities, Concurrent RTE delivers a sophisticated foundational building block for your Service-oriented Infrastructure. Examples of other building-block components include Xen virtualization, fabric-based quality of service, cluster file systems and Distributed Shared Memory. As these additional components are refined and optimized, you will be able to deploy them on Concurrent RTE. The following sections explore the advantages of deploying a real-time kernel with the other building blocks of a Service-oriented Infrastructure.

### *Real-time Distributed Shared Memory (RT-DSM)*

Many proprietary applications leverage Distributed Shared Memory (DSM) capabilities to implement highly efficient clustering solutions. These DSM implementations typically take advantage of high-speed interconnects such as Infiniband (IB). However, one of the primary problems with DSM implementations is that they cannot keep all nodes coherent with one another when one node updates locations in the shared memory segment. For certain types of applications, like financial trading programs, a consistent view of DSM is critical.

With Concurrent RTE, you can optimize Distributed Shared Memory based on Infiniband. You can even carve out dedicated real-time lanes from an IB connection. A dedicated lane provides guaranteed bandwidth in the connection between nodes. A dedicated lane also provides better latency guarantees, so you'll know exactly when all your DSM-attached nodes will be updated. Finally, this feature allows multiple nodes to share a coherent view of the shared memory.

In the real world, it's difficult to move an application from one node to another without data loss. This is because process migration requires you to move huge amounts of application memory from one node to another. By using Distributed Shared Memory, you significantly reduce the amount of memory you have to transfer, since not all memory must reside on the node where an application runs.

If your applications leverage Distributed Shared Memory and also require persistent storage, you would greatly benefit by integrating a cluster file system along with the DSM module. Concurrent RTE enhances both of these capabilities.

**With Concurrent RTE, you can optimize Distributed Shared Memory based on Infiniband. You can even carve out dedicated real-time lanes from an IB connection.**

It's historically been difficult to build efficient virtual environments because they add overhead to the executing system.

**Your applications get guaranteed quality of service (QoS) from the real-time features in Concurrent RTE. The ability to not be interrupted is desirable—if not essential—for many of today's applications.**

### *Real-time Xen Hypervisor*

It's historically been difficult to build efficient virtual environments because they add overhead to the executing system, especially when compared against native OS technologies. An exciting open source project, Xen, multiplexes resources at the granularity of an entire OS, as opposed to process-level multiplexing. (This GuestOS is also called a container.) All user-level applications run within a GuestOS and each GuestOS may be dedicated to a single application or it may deploy multiple applications. The Xen Hypervisor is responsible for context switching GuestOSes. This process involves not only saving the state of the GuestOS, but also all the applications running within the GuestOS.

Recent enhancements to the Xen Hypervisor have made it much more sensitive to scheduling priorities, memory demand, and network and disk traffic. A latency-aware Hypervisor enables GuestOSes to run in a virtual machine environment without compromising overall system performance. In addition, the GuestOSes can possess real-time priorities, enabling virtual containers (and all applications in those containers) to run without normal system disruptions (such as interrupts and daemons). Finally, a real-time GuestOS delivers granular control over system level resources (such as the processor, memory and network) in a virtualized environment.

### *Synchronized Real-time Timers*

Time-critical applications that run in a clustered environment will likely need time-stamping capabilities. A normal clustered environment provides no guarantees of clock synchronization across multiple cluster nodes. In fact, an application could stamp completely unsynchronized times if it runs on different parts of the cluster (due to clusterwide process migration). If you're running an application of this kind, it would greatly benefit from system clocks that are fully synchronized across an entire cluster of machines.

### *Real-time Cluster File Systems (RT_CFS)*

Cluster file systems rely on synchronization mechanisms, typically distributed lock management (DLM), to guarantee data integrity across the cluster. You can also use DLM to synchronize access to metadata objects for the file system and provide POSIX file-locking capabilities for user-level data.

Software layers on top of each OS facilitate intra-cluster communication and manage services and resources. Specifically, a cluster file system provides each node in the cluster with access to the same storage partitions that reside on the Storage Area Network (SAN). This enables applications on each node to access the same files. Using Concurrent RTE, your cluster file systems can take advantage of deterministic behavior throughout the cluster, resulting in overall performance improvements.

### *Fabric Virtualization (QoS-based Clusters)*

Your applications get guaranteed quality of service (QoS) from the real-time features in Concurrent RTE. The ability to not be interrupted is desirable—if not essential—for many of today's applications. In fact, many of today's computing-intensive traditional applications would greatly benefit from being able to complete tasks without scheduler-driven interruptions.

It is our goal to enrich the real-time capabilities currently provided within a system and offer these capabilities across the distributed computing network. Real-time functionality must extend across the connecting fabric

in such a manner that the fabric respects the issuing process's priority across the network. This is the primary attribute of a QoS-based cluster.

## Virtual Computing Platform

The virtual computing platform will be to distributed computing what the automobile was to the horse and carriage. If your data centers are overwhelmed, and you want to consolidate siloed application architectures and widely underutilized resources, it's time to work toward virtual computing.

Virtual computing is a revolutionary technology that will integrate commodity solutions. However, today's businesses must first achieve a successful mix of proprietary and commodity solutions before they can transition to the virtualized data center. The recent economic downturn emphasized that over-provisioned and underutilized data centers are unaffordable and drive unnecessary capital spending. By adopting platforms based on commodity components, you will be much more likely to maximize your return on investment.

We anticipate that the real-time building blocks at the core of the operating system will eventually extend out to the fabric and create a QoS-based distributed computing environment. This on-demand environment will be highly sensitive to latency requirements in a network-computing infrastructure and serve as the required disruptive technology that enables the virtual computing platform. In this setting, agility will become one of your most attractive competitive advantages, as you provide customers with dynamic and consistent results.

## Integrated Services

The Concurrent RTE solution offers a complete suite of services delivered by an integrated Novell and Concurrent team.

### *Support Services*

Concurrent RTE requires extensive modifications to the Linux kernel. When you deploy mission-critical applications on this real-time infrastructure, you'll want to keep your kernel environment up to date with the latest releases. An integrated services team from Novell and Concurrent is ready to provide you with world-class support for this offering. Novell front-line support personnel will assess your issue and engage the Concurrent real-time support infrastructure as appropriate.

### *Professional Services*

The following professional service offerings are also available for Concurrent RTE:

- **Deployment services.** *Service engineers will help you deploy your real-time environment. This offering includes system administrative support as well as real-time application and system tuning.*
- **Custom image builds.** *Service engineers can help you create custom real-time images for client environments deployed across a wide variety of systems.*
- **Application migration and real-time enablement.** *These services help you migrate proprietary applications to the real-time environment. This includes application enablement, so you can take advantage of the many programmatic features offered in Concurrent RTE.*
- **Training.** *Our services organization offers many training courses around real-time deployments. You can opt for a Web-based seminar or choose hands-on training sessions that focus on deploying real-time systems and writing real-time applications.*

## Summary

Concurrent RTE delivers a unified environment for workloads across different industry segments. It provides high availability, real-time processing and predictability for your individual systems and all the components that connect them.

**The following attributes are crucial to a commodity platform:**

- **Lower cost**
- **Higher performance**
- **Modular components**
- **Open standards**

An integrated services team from Novell and Concurrent is ready to provide you with world-class support for Concurrent RTE.

www.novell.com

With Concurrent RTE as your foundation, you can create a Service-oriented Infrastructure, rapidly integrating other technologies and standards that support agile network-based computing. Looking ahead, you'll be able to automate and manage your software services and resources as commodity components and then deploy them freely on a virtual-computing platform. Finally, your ability to quickly respond to system-level events and high-priority processes will give you a network that carries quality of service guarantees across the fabric.

### *About the Authors*

Steve Brosky is Chief Scientist of Real-Time Products for Concurrent Computer Corporation, a leading provider of high-performance, real-time Linux software and solutions for commercial and government markets. As Chief Scientist, Brosky determines future development strategies, defines new software projects and actively participates in the design and development of new software products for real-time systems. He is also responsible for presenting Concurrent's real-time products to customers, using these forums to better adapt the company's product line to customer needs.

Moiz Kohari is Principal Technologist at Novell, leading the SUSE Lab real-time and clustering initiatives. Kohari is deeply engaged in Novell efforts regarding fabric virtualization and quality of service offerings, including Carrier Grade Linux. Before joining Novell, Kohari was Founder and CEO of Mission Critical Linux, an organization that pioneered sub-second failover and clustering technologies on Linux. He has 20 years' experience designing, implementing and supporting operating systems. Kohari was a member of Digital/Compaq's Tru64 UNIX custom development and support team, concentrating on enterprise computing and multiprocessor system performance.

Contact your local Novell Solutions Provider, or call Novell at:

1 888 321 4272 U.S./Canada
1 801 861 4272 Worldwide
1 801 861 8473 Facsimile

For more information on Concurrent RTE, visit:
www.novell.com

**Novell, Inc.**
404 Wyman Street
Waltham, MA 02451 USA

concurrent

Novell