Information Center for Linux

Best practices for tuning system latency

Information Center for Linux

Best practices for tuning system latency

Note

Before using this information and the product it supports, read the information in "Notices" on page 15.

First Edition (March 2011)

Contents

Best practices for tuning system latency

You can tune your applications, operating system (OS), and hardware environments for low latency.

Note that low latency, as with Real-Time, does not always result in higher throughput. Throughput is often sacrificed for determinism and low latency. This document applies to the application layer, the OS layer, and hardware-specific BIOS options. The portions of this document relevant to the user environment will vary depending on the hardware platform, the Linux distribution involved, and the application requirements.

Scope

This document is written to be generically useful, and might also contain information that is unique to a particular Linux Distribution (Red Hat Enterprise Linux and SUSE), or a particular hardware platform.

You can find information specific to running a Real-Time (PREEMPT_RT) kernel at the end of this document.

Scalability

Carefully consider systems with more than 16 processors as there might be scalability issues with locking and scheduling events that are not seen with smaller system configurations.

Workloads requiring coordination between threads, or sharing the use of I/O resources, are potentially most affected.

Long-running, processor-intensive workloads that do not significantly deal with locks or thread scheduling issues are generally less affected by scalability issues.

Application tuning

Several methods and tools are available for you to tune applications on your system.

System memory and application memory locking

Ensure that the platform has sufficient memory to accommodate the application needs.

Ideally, all of the application pages and their data are permanently pinned to memory with system calls like mlockall() to avoid latency hits due to paging. It is normal for page faults to occur during an application start up.

One easy way to avoid page faulting is to pin all application pages into memory with mlock, for example:

```
mlockall(MCL_CURRENT|MCL_FUTURE);
```

Page faults should be avoided when latency-sensitive parts of the application are running, including allocation of memory and I/O operations in particular. Some behaviors to avoid include:

- Creating new threads
- Allocation of memory (malloc)
- Loading dynamic libraries (dlopen)
- Disk or other I/O related activity

For more information about the mlock command, see the mlock man page.

Processor binding

On systems with many processors, binding application threads to specific processors or sets of processors can reduce scheduler overhead, eliminate system time spent on migrating tasks between processors, and help ensure the application data remains in the caches.

On systems with NUMA characteristics, be aware of how the system memory is allocated to those NUMA nodes. Applications bound to processors in a specific NUMA node should be bound to memory in that same NUMA node.

Be aware that binding applications to processors can also lead to lessened system use, as there might be scenarios where some processors are idle and applications can not be migrated to those processors due to how the applications are bound.

Processor binding can be achieved in different ways:

- Using taskset:
  - To retrieve the current processor mask, use the `taskset -p <pid#>` command as shown in the following example:

    ```
    taskset -p 4612
    ```

    The affinity mask for pid 4612 is 2.
  - To set a new processor mask, use the `taskset -p <cpu_mask> <pid#>` command as shown in the following example:

    ```
    taskset -p 0x01 4612
    ```

    The affinity mask for pid 4612 is changed from 2 to 1.
- Using cpusets and control groups

  Control groups provide a mechanism for aggregating and partitioning sets of tasks, and all their future children, into hierarchical groups with specialized behavior. Cpusets use the control group subsystem to assign sets of processors and memory nodes to a set of tasks, constraining the task to only the resources provided by that cpuset. Use numactl –hardware to determine the characteristics of the system, particularly the correlation between processors and memory nodes.

  The libcgroup (libcg) package contains APIs and tools that provide you with greater control over the control group filesystem. The following example uses the kernel interfaces directly:

  1. To initially set up the control group filesystem:

     ```
     mkdir /dev/cpuset
     mount -t cpuset none /dev/cpuset
     ```
  2. Create a cpuset, add cpus, and add memory nodes:

     ```
     mkdir /dev/cpuset/myapp_cpuset
     echo 12,13,14,15 > /dev/cpuset/myapp_cpuset/cpus
     echo 1 > /dev/cpuset/myapp_cpuset/mems
     ```
  3. Move the appropriate tasks into the taskset:

     ```
     for pid in `pidof myapp` ; do
     echo $pid > /dev/cpuset/myapp_cpuset/tasks ; done
     ```

  The target applications are now confined to the processors and memory nodes as identified by the cpuset.

  For more information about the numactl command, run the man numactl command.

  For more information about the taskset command, run the man taskset command.

  For more information about the capabilities command, run the man capabilities command.

  Tip:

1. In the above examples, the referenced cpu_mask is a bitmask indicating processors with the lowest order bit corresponding to the first logical processor. For example:

   0x00000001 is processor #0

   0x00000003 is processors #0 and #1

   0xFFFFFFFF is all processors (#0 through #31)

   The taskset command also allows the use of a -c parameter to indicate a processor list. This list syntax allows ranges and might be more intuitive. For example:

   0,2 is processor #0 and #2

   0-3,7-31 is processors #0 through #3 and #7 through #31

2. Set the CAP_SYS_NICE capability so that a user can change the processor affinity or scheduling attributes of a process. If this capability is not set, the chrt and taskset utilities return an `Operation not permitted` error.

For more information about processor binding, see "CPU shielding using /proc and /dev/cpuset" at https://rt.wiki.kernel.org/index.php/CPU_shielding_using_/proc_and_/dev/cpuset.

To prevent IRQ activity on a particular processor, you can use IRQ shielding to prevent certain interrupts from being serviced on particular processors. This is useful in scenarios where an important application thread or task should not be interrupted.

To perform this shielding:

1. Disable the irqbalance daemon:

   (temporary) `/etc/init.d/irqbalance stop`

   (permanent) `chkconfig irqbalance off`

   Tip: The chkconfig step alone does not stop any currently running irqbalance daemons.

2. Set the processor affinity for the IRQ:

   ```
   echo <hex_mask> > /proc/irq/<irq_number>/smp_affinity
   ```

3. Verify the current affinity:

   ```
   cat /proc/irq/<irq_number>/smp_affinity
   ```

Remember: The displayed smp_affinity value is updated the next time an interrupt is serviced, so you might not see this change take effect immediately.

For more information about IRQ shielding, see the `Documentation/IRQ-affinity.txt` file.

For more information about cpusets, see:

- the `Documentation/cgroups/cgroups.txt` file.
- the `Documentation/cgroups/cpusets.txt` file.
- the Documentation/cpusets.txt article on LWN.net at http://lwn.net/Articles/127936/.

Scheduling policies

The scheduler decides which process is run next by a processor, and there are several scheduling policies available that can be set for a process or application to help determine how the scheduler prioritizes which process runs next.

Some of these scheduling policies are:

SCHED_OTHER
: the default scheduling policy for all processes that do not require special static priority real-time mechanisms

SCHED_FIFO (First-In, First-Out)
offers the most powerful method for threads that are latency sensitive. Processes in the SCHED_FIFO class run on a processor for as long as they remain runnable, subject only to the needs of higher-priority Real-Time processes. SCHED_FIFO processes that become runnable immediately preempt any currently running SCHED_OTHER process or any lower priority SCHED_FIFO process.

SCHED_RR (Round Robin)
similar to SCHED_FIFO, except that each process is limited to a maximum time quantum. When a SCHED_RR process has been running for a time period as long as (or longer than) the quantum, it is placed at the end of its list for its priority.

You can set the scheduling policy in code as shown:

```
struct sched_param sp;
sp.sched_priority = 77;
if (sched_setscheduler(0, SCHED_FIFO, &sp) != 0)
perror("sched_setscheduler");
```

You can set the policy for a new or already running process on the command line with the chrt command as shown:

```
chrt -f 77 <cmd>
chrt -f -p 77 <PID>
```

For more information about scheduling policies, see:

- the sched_setscheduler man page.
- the chrt man page.
- the `Documentation/scheduler/sched-design-CFS.txt` file.
- the SCHED_FIFO and realtime throttling article at http://lwn.net/Articles/296419/.

Thread synchronization

You can synchronize threads to prevent priority inversion whenever possible.

For example, ensure that any high-priority, latency-sensitive threads, such as those running with the SCHED_FIFO policy, do not have to synchronize a shared mutex with lower-priority threads. In particular, any thread with a SCHED_OTHER policy is of lower priority than a thread with a SCHED_FIFO policy.

Use the PTHREAD_PRIO_INHERIT protocol attribute for mutex operations as this allows the current owner of the mutex to run at the higher priority of any thread waiting on the mutex. For example:

```
pthread_mutexattr_setprotocol(&attr, PTHREAD_PRIO_INHERIT);
```

For more information about the pthread_mutexattr_setprotocol command, see the pthread_mutexattr_setprotocol man page.

Measuring time

To measure time reliably across processors, always use CLOCK_MONOTONIC.

CLOCK_MONOTONIC always increases linearly with time. CLOCK_REALTIME, however, can change depending on changes to system time, for example, on a system running a Network Time Protocol daemon (ntpd). The following example demonstrates how to measure time using CLOCK_MONOTONIC:

```
clock_gettime(CLOCK_MONOTONIC, &start);
do_work();
clock_gettime(CLOCK_MONOTONIC, &stop);
```

On a platform that has high-resolution timers (HRT) enabled, measure the resolution of the clock to verify the accuracy. Use the clock_getres call to confirm this, as follows:

```
clock_getres(CLOCK_MONOTONIC, &tp);
```

Important: HRTs require both hardware support and enablement within the operating system. SUSE SLES 11.* and Red Hat Enterprise Linux 6.* distributions contain operating system support for HRTs.

The TSC clocksource is not guaranteed to be synchronized across processors, and is not a good time source for relative time measurements (unless the measured time events are within a thread that is bound to a particular processor).

Avoid using sched_yield() as it rarely leads to deterministic behavior. In a scenario where sched_yield() is called from a high priority SCHED_FIFO process, if all other runnable threads are of lower priority, the high priority thread immediately resumes on the processor, negating the intent of calling sched_yield().

For more information about the sched_yield command, see the sched_yield command man page.

Avoid making system calls that result in I/O during latency-sensitive paths. I/O devices are generally not considered low-latency-safe, as they may be bound by circumstances entirely outside the control of the operating system. Sometimes it is not obvious whether a particular call can lead to disk I/O. For example, a call to getpwuid() (get password file entry) can lead to opening files on the disk if the nscd daemon is not running.

Optimize the application's use of timers as they can incur kernel overhead and can increase jitter.

For more information about the clock_getres command, see the clock_getres man page.

Operating system tuning

Several methods and tools are available for you to tune your system.

Services to disable

Stop any tasks or daemons that are unnecessary for the target environment.

Examples of daemons that might not be necessary for a server environment include:

cupsd
selinux
sendmail
gpm
cron jobs
pcscd

Note: This is an arbitrary list of services. The actual list depends on the requirements of the application and the environment.

To list running or configured services, you can run commands similar to the following:

- (Red Hat) `chkconfig --list`
- (SUSE) `yast2 runlevel summary`

To disable services, you can run commands similar to the following:

- (Red Hat) `chkconfig cupsd off`
- (SUSE) `yast2 runlevel delete service=cups`

Choose the proper runlevel. If you do not require the X11/x-server, check that your system is set for initlevel 3, or that X11 is otherwise disabled.

Check if HRT clocksources are available, and use them if possible. ACPI_PM and HPET timers are both HRT clocksources. Available and current clocksources can be checked with sysfs on a running system, as shown:

```
cat /sys/devices/system/clocksource/clocksource0/available_clocksource
cat /sys/devices/system/clocksource/clocksource0/current_clocksource
```

High resolution clock sources might not be available on all platforms, and such systems are not recommended for running latency-sensitive applications. If you still need to run on a system without HRTs, be aware that fine-grained time resolution and responsiveness will not be available.

User authority for operating system real-time capabilities

Low-latency-sensitive applications will try to pin all their memory and use (through chrt or sched_setscheduler()) real-time priorities for application threads.

To enable this capability, create a group with these special privileges and add the required users to this group. Typical values for this privileged group can be set in a configuration file as follows:

```
TIMEOUT=unlimited            # in minutes
RT_PRIORITY=100
NICE=-20
MEMLOCK=unlimited

cat >>/etc/security/limits.conf <<EOF
@realtime    soft  cpu       $TIMEOUT
@realtime    -    rtprio     $RT_PRIORITY
@realtime    -    nice      $NICE
@realtime    -    memlock     $MEMLOCK
EOF
```

User profiles can then be added to the realtime group with usermod:

```
usermod -g realtime <userid>
```

Users must sign out and then sign back into the system for these changes to take effect.

For more information about the `limits.conf` file, see the `limits.conf` man page.

Tuning SCHED_FIFO runtime limits

By default, SCHED_FIFO tasks are limited to 95% of the total system processor time, enforced on a per-second period. The remainder of that time, 5%, is reserved for SCHED_OTHER tasks.

These defaults are intended to prevent a run-away real-time task from locking up the system. These settings can be adjusted with procfs:

```
cat /proc/sys/kernel/sched_rt_runtime_us
   950000
 echo 980000 > /proc/sys/kernel/sched_rt_runtime_us
```

Important:

1. The sched_rt_runtime_us tuning parameter is available in Red Hat Enterprise Linux 6.* and SUSE SLES 11.* releases.
2. The 980000 sample shown above is for example purposes only and is not a specific recommendation.

For more information about the SCHED_FIFO runtime limit, see the `Documentation/scheduler/sched-rt-group.txt` file.

Tuning network latency

Network and storage activity can be tuned, but generally are not considered safe (or guaranteed) for low latency systems.

This is true of most peripheral and I/O devices in the system. With the following steps, make sure applications perform non-blocking I/O wherever possible to allow continued progress:

1. Enable the tcp_low_latency kernel parameter and set the sysctl variable net.ipv4.tcp_low_latency to a value of 1 in /etc/sysctl.conf (or equivalent) to take effect across system restarts. This can also be set on a running system with proc. For example:

   (with sysctl.conf) `net.ipv4.tcp_low_latency=1`

   (with proc) `echo 1 > /proc/sys/net/ipv4/tcp_low_latency`
2. Increase the interval between scheduled system route flushes. For example:

   (with sysctl.conf) `net.ipv4.ipfrag_secret_interval = 6000`

   (with proc) `echo 6000 >> /proc/sys/net/ipv4/route/secret_interval`

   Note: Monitor the system route flushing intervals carefully to ensure stale routes don't cause network issues.
3. For better performance, increase the default and max UDP and TCP socket buffer sizes as well as network queue lengths. This is especially useful when the application reading data from a socket buffer is not scheduled fast enough because the kernel will drop incoming packets when the buffers are full.

   The core socket read buffer sizes (default and max):

   (with sysctl.conf) `net.core.rmem_default = 8388608`

   (with sysctl.conf) `net.core.rmem_max = 16777216`

   (with proc) `/proc/sys/net/core/rmem_default`

   (with proc) `/proc/sys/net/core/rmem_max`

   The write (send) core system sizes:

   (with sysctl.conf) `net.core.wmem_default = 8388608`

   (with sysctl.conf) `net.core.wmem_max = 16777216`

   (with proc) `/proc/sys/net/core/wmem_default`

   (with proc) `/proc/sys/net/core/wmem_max`

   The queue lengths and related parameters:

   (with sysctl.conf) `net.core.netdev_max_backlog = 20000`

   (with sysctl.conf) `net.core.optmem_max = 25165824`

   (with proc) `/proc/sys/core/netdev_max_backlog`

   (with proc) `/proc/sys/core/optmem_max`

For more information about the following commands, see their man pages:

- sysctl
- sysctl.conf
- tcp <tcp_low_latency>, <rmem_default>, and so on

Efficient IP address multicasting

An application that subscribes to a multicast group using a socket that is bound to the wildcard IP address (INADDR_ANY) receives all the multicast packets that come in to the host and are delivered to user space by the kernel. The application must process the incoming data and discard unwanted packets.

The wake-ups and processing of those unwanted packets is unnecessary, and of no benefit to the application workload.

If you want the application to receive packets only from the multicast group it has subscribed to, while still binding to a wildcard address (INADDR_ANY), set the IP_MULTICAST_ALL socket option.

The IP_MULTICAST_ALL option is a boolean integer set to off by default. Enabling this option ensures that only the packets from multicast groups the socket is subscribed to will be delivered.

This change does require that the applications explicitly subscribe to each multicast group that you want them to receive data from.

Usage:

```
opt = 0;
     setsockopt(sockfd, IPPROTO_IP, IP_MULTICAST_ALL, &opt, sizeof(opt));
```

The benefit of this change is dependent on the frequency and volume with which the unwanted packets are received. The IP_MULTICAST_ALL option is available in SUSE SLES 11.1 and Red Hat Enterprise Linux 6.* releases.

For more information about the setsockopt command, see the setsockopt man page.

Hardware, BIOS, and firmware settings

Some system firmware (either BIOS or UEFI) settings can affect determinism of the running operating system. You can change these settings through the Configuration/Setup Utility program (BIOS) or the Setup utility (UEFI).

Be aware that firmware updates might change the default settings for some UEFI options. For more information about the firmware updates and settings, see the firmware change logs.

Settings that can be changed are the following:

Operating modes
: Provides an initial set of processor and memory settings based on generic profiles. The default value is `Custom`, which provides a mix of performance and power savings settings. It is recommended that this be set to `Performance`.

TurboMode
: Provides the ability to overclock a set of processor cores based on the power and thermal headroom in the environment. This option is enabled by default, but it is recommended that you disable this setting for environments with latency-sensitive or clock frequency-sensitive applications.

TurboBoost
: Specifies the type of TurboMode algorithm to use when determining whether to overclock the cores in the system. This setting is not applicable where TurboMode is disabled.

Processor performance states (voltage and frequency scaling)
: Allows for an automatic adjustment of clock frequencies and core voltages dependent on the processor use, with the intent of reducing power consumption. For latency-sensitive applications, it is recommended that this be set to `Max Performance`.

Processor sleep states (C-States)
: Microprocessors are able to enter low-power modes or sleep states. These sleep states can cause unbounded latencies in real-time applications. To avoid these problems, disable sleep states and other low-power states. Depending on the processor package in your server, an option such as Processor Performance States, Processor C-States, or C1 Enhanced Mode might be available in the BIOS or the UEFI. Disable the relevant option to increase the overall determinism of real-time applications that are running on the server.

Hyper-Threading (HT)
Turn off HT unless the multi-threaded modes are known to help the workload. The hardware threading features can result in an increase in application latencies due to increased cache contention pressure, adversely affecting determinism and low latency in exchange for increased throughput. Results will vary depending on the workload.

Processor data prefetchers
When enabled (by default), attempts to predictively retrieve information from main memory into the processor's L1 or L2 cache, allowing better processor use. Depending on the data patterns, this might be detrimental if incorrect information is cached, which then requires flushes and subsequent loads of different data. This option is enabled by default on platforms that support it, and the impact on latencies depend on the workload environment.

Data reuse
Allows the processor to retain frequently used cache lines at the expense of control signals between the processor and cache. Enabling this setting increases the overall power and thermal draw. This setting is recommended for HPC applications. The benefit for this setting on a low latency workload can vary.

Memory verification (PatrolScrub and DemandScrub)
Determine how and when system memory is scrubbed for errors. Default settings are disabled and enabled, respectively. Consider enabling these options for production systems because they will aid in maintaining the resiliency of the memory subsystem. Where platform redundancy is gained through a distributed computing model, these options can be disabled to gain additional performance.

Tools for tuning and evaluating system latencies

You can use tuning and evaluating tools to find code hot-spots or measure system and application latencies.

Three helpful tools are described as follows:

trace-cmd
A wrapper allowing easier use of the ftrace tooling built into the kernel. This is helpful in measuring latencies as seen by the kernel. The ftrace options are enabled and available in Red Hat Enterprise Linux 6.*.

For more information about the trace-cmd, see trace-cmd - command line reader for ftrace at http://lwn.net/Articles/341902/.

rt-tests
Includes an assortment of tests useful for measuring system latencies and application latencies.

For more information about rt-tests, see:

- rt-tests at http://www.pengutronix.de/software/rt-tests/index_en.html.
- rt-tests index at http://www.kernel.org/pub/linux/kernel/people/clrkwllms/rt-tests/.

Oprofile
A system-wide profiler for Linux systems that allows for the analysis of system performance, the identification of code hot-spots, and contains support for hardware performance counters. Oprofile tools are included in both Red Hat Enterprise Linux and SUSE distributions.

For more information about Oprofile tools, see Oprofile at http://oprofile.sourceforge.net/about/.

The PREEMPT_RT patch set

The PREEEMPT_RT patchset contains features not yet incorporated into upstream kernel sources that might be helpful for achieving better low-latency behavior in some environments.

These features are available with several Linux distributor products, as well as by building an upstream kernel manually. Features of the PREEEMPT_RT patchset include:

- Preemptible spinlocks (in-kernel locking sections) through reimplementation with rtmutexes. Portions of the kernel that are not safe for preemption are converted to raw_ spinlocks.
- Preemptible critical sections.
- Priority inheritance for spinlocks and semaphores.
- Interrupt handlers (soft) converted to preemptible kernel threads.

Both Red Hat and SUSE have the following real-time products available:

- Red Hat Messaging, Real-Time, Grid (MRG)
- SUSE Linux Enterprise Real-Time (SLERT)

A PREEMPT_RT kernel can also be built manually using upstream kernel sources and readily available patches.

Tips for PREEMPT_RT patch set-enabled systems

Be aware of how issues like scalability and threaded interrupts can affect systems that are enabled with the PREEMPT_RT patch set.

Scalability, thread priorities, and System Management Interrupts (SMIs) can affect your system if the PREEMPT_RT patch set is enabled:

Scalability
For a real-time kernel, scalability issues might be seen at lower processor counts than with non real-time kernels, due to changes in locking in the kernel. These locking-related changes improve determinism but result in increased pressure (bottlenecks) on any contended locks.

Thread priorities
On a system running the PREEMPT_RT patchset, it is possible for a system designer to change the priorities of both kernel threads and userspace application threads.

The ability to control the priority of kernel threads allows much greater control over the system behavior, and should be done carefully. A higher priority application with a busy loop can starve out a lower priority kernel thread, which can lead to loss of data due to a throttled network transmit or receive threads, or system lockups if an important kernel thread is stuck behind a high priority userspace thread which is itself stuck in a loop.

In general, set the priorities of application threads so that they remain lower than that of the kernel threads. If the application is missing deadlines due to the interference of kernel threads, use IRQ shielding in conjunction with processor binding.

Threaded interrupts and IRQ threads
The real-time kernel runs most interrupts and softirqs as threads. Hence, it is possible to list them using the ps command and change their priorities using the chrt command. Take care when you are deciding the priority values for these kernel threads as incorrectly configured priorities could potentially hang the system. Linux distributors that ship a real-time kernel usually provide a tool that makes it easy to configure the priorities of these threads. Red Hat's MRG ships a tool called rtctl, SUSE's SLERT ships a tool called set_kthread_prio.

IRQ threads are set by default to run at a higher priority than most user threads. One way to ensure that the IRQ threads don't interrupt application threads is to bind the application and the IRQs to different sets of processors. This can be done as follows:

1. Turn off the irqbalance daemon if it is running, as shown:
   ```
   chkconfig --level 3 irqbalance off
   ```
2. Use either the taskset command or containers or cpusets to bind the application to a processor or set of processors.

3. Check the current set of IRQ or processor affinity bindings, as shown, where *xx* is the IRQ number:

   ```
   cat /proc/irq/xx/smp_affinity (xx = IRQ number)
   ```

4. Change the affinity using commands similar to the following example, where *xx* is the IRQ number:

   ```
   /bin/echo custom_cpu_mask > /proc/irq/xx/smp_affinity
   ```

You can also use operating system tools and libraries, like libcg, for creating containers and cpusets, and distribution-specific tools like tuna (Red Hat) or cset (SUSE) for managing the cpusets and thread priorities.

IBM Premium Real-Time Mode for handling SMIs
SMIs can occur periodically (for temperature or voltage control related events), or infrequently (for memory errors or thermal warning events). These SMIs can lead to non-desirable behavior as the BIOS or firmware might take a long time to process these interrupts, entirely outside the control of the operating system. On some hardware platforms, the IBM premium real-time mode (using the ibm-prtmd daemon in conjunction with BIOS modifications) disables all non-fatal management interrupts to enable real-time processes to run in a more deterministic environment.

Tools for PREEMPT_RT-based kernels

You can use a combination of the subsystem, services, and tools to set thread priorities or groups of processes

Four helpful tools are described as follows:

perf The Performance Events subsystem allows event enumeration, reporting, logging, monitoring, and analysis. The core of this tool is built into the Linux kernel, with some userspace portions required for use.

For more information about perf, see the `tools/perf/Documentation/perf.txt` file.

(SUSE) set_kthread_prio
This service uses the `/etc/set_kthread_prio.conf` file to set the priorities and scheduling policy of kernel threads. This service must be restarted to pick up changes to the configuration file.

For more information about set_kthread_prio, see SLERT at http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/index.jsp?topic=/liaai/realtime/liaairtrtctlslert.htm.

(Red Hat) rtctl
This service sets the scheduling policies and priorities of groups of processes as defined in the `/etc/rtgroups` file. This service must be restarted for changes to take affect.

For more information about rtctl, see:

- the RHEL-RT Wiki: RHEL-RT RTCTL article at http://rt.et.redhat.com/page/RHEL-RT_RTCTL.
- the MRG Realtime at http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/index.jsp?topic=/liaai/realtime/liaairtrtctl.htm

Tuna This tool provides both graphical and command line interfaces for changing scheduler and IRQ tunables at whole processor and per-thread or IRQ levels.

For more information about the Tuna GUI, see RHEL-RT Wiki: Tuna GUI at http://rt.et.redhat.com/wiki/index.php?title=Tuna_GUI.

For more information about PREEMPT_RT tools, see Tuna User Guide at http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_MRG/1.3/html/Tuna_User_Guide/index.html.

Configuration options for PREEMPT_RT-based kernels

The PREEMPT_RT kernel patch set requires these configuration options to be enabled to use the real-time capabilities.

Red Hat MRG and SUSE SLERT distribution kernels have these options enabled by default:

- CONFIG_PREEMPT_RT
- CONFIG_HIGH_RES_TIMERS
- CONFIG_HZ (set to 1000)
- CONFIG_TRACING

Additional tracing options are available for specific subject areas, function tracing, scheduler tracing, IRQ tracing, and so on.

Important: Set additional tracing options to OFF in production environments as these options could introduce additional overhead and latencies.

In a production environment, it is recommended that these options are set to OFF:

- CONFIG_DEBUG_PREEMPT
- CONFIG_DEBUG_RT_MUTEXES

Appendix. Related information

You can find additional information about the processes and tools described in these best practices.

- Using rtctl to assign real-time priorities to process groups

  http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/topic/liaai/realtime/liaairtkernalrtp.htm
- SUSE Linux Enterprise Real Time Extension

  http://www.novell.com/products/realtime10/techspecs.html
- SUSE Linux Enterprise Real Time 10 SP1 Quick Start

  http://www.novell.com/documentation/slert/pdfdoc/sle-rt_quick/sle-rt_quick.pdf
- Realtime Tuning Guide

  http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_MRG/1.3/html/Realtime_Tuning_Guide/index.html

Notices

This information was developed for products and services offered in the U.S.A.

IBM® may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Dept. LRAS/Bldg. 903
11501 Burnet Road
Austin, TX 78758-3400
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to the manufacturer, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. The manufacturer, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

CODE LICENSE AND DISCLAIMER INFORMATION:

The manufacturer grants you a nonexclusive copyright license to use all programming code examples from which you can generate similar function tailored to your own specific needs.

SUBJECT TO ANY STATUTORY WARRANTIES WHICH CANNOT BE EXCLUDED, THE MANUFACTURER, ITS PROGRAM DEVELOPERS AND SUPPLIERS MAKE NO WARRANTIES OR CONDITIONS EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT, REGARDING THE PROGRAM OR TECHNICAL SUPPORT, IF ANY.

UNDER NO CIRCUMSTANCES IS THE MANUFACTURER, ITS PROGRAM DEVELOPERS OR SUPPLIERS LIABLE FOR ANY OF THE FOLLOWING, EVEN IF INFORMED OF THEIR POSSIBILITY:

1. LOSS OF, OR DAMAGE TO, DATA;
2. SPECIAL, INCIDENTAL, OR INDIRECT DAMAGES, OR FOR ANY ECONOMIC CONSEQUENTIAL DAMAGES; OR
3. LOST PROFITS, BUSINESS, REVENUE, GOODWILL, OR ANTICIPATED SAVINGS.

SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF DIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, SO SOME OR ALL OF THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information in softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® and ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java and all Java-based trademarks and logos are registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Printed in USA