

Linux Configuration V6.6

0.1 Einführung

Dieses Dokument dient zur Beschreibung von diversen Einstellungen bei der Konfiguration mittels `make menuconfig` unter Linux.

Es wird nicht näher darauf eingegangen, wie der Kernel kompiliert wird oder welche Voreinstellungen, Programme etc. zum Kompilieren benötigt werden.

Zu Beginn der jeweiligen Konfigurationszeile wird der Standardwert (Default) angezeigt. Mein Vorschlag folgt danach.

Z. B. bei `CONFIG_WERROR [=n] [Y]`

Hier ist der Standardwert ein Nein `[n]`, meine persönliche Einstellung ein Ja `[Y]`.

@KW4NZ, Thomas Kuschel

Wenn Sie Tippfehler finden oder Korrekturen wünschen, dann schicken Sie dies mit Erläuterungen und dem Hinweis auf die obenstehende Version V6.6 an: oe1tkk@gmail.com

1 General setup →

1.1 Compile also drivers which will not load

`CONFIG_COMPILE_TEST [=n] [ ]`

Kompilieren Sie auch Treiber, die nicht geladen werden können

Einige Treiber können auf einer anderen Plattform kompiliert werden als auf der, für die sie gedacht sind. Obwohl sie dort nicht geladen werden können (oder selbst wenn sie geladen werden können, können sie aufgrund fehlender Hardware-Unterstützung nicht verwendet werden), möchten Entwickler, im Gegensatz zu Distributoren, solche Treiber vielleicht trotzdem kompilieren und testen.

1.2 Compile the kernel with warnings as errors

`CONFIG_WERROR [=n] [Y]`

Den Kernel mit Fehlermeldungen bei Warnungen kompilieren

Ein Build sollte keine Compiler-Warnungen ausgeben, dies aktiviert die Flags `'-Werror'` (für C) und `'-Dwarnings'` (für Rust) um diese Regel standardmäßig zu setzen. Bestimmte Warnungen von anderen Tools z.B. der Linker könnte mit dieser Option Fehler generieren. Deaktivieren ist sinnvoll, wenn Sie einen neuen (oder sehr alten) Compiler bzw. Linker mit seltenen, ungewöhnlichen Warnungen haben. Haben Sie auf Ihrer Architektur Probleme, dann müssen Sie diese Konfiguration deaktivieren, um den Kernel erfolgreich zu bauen. Im Zweifelsfall sagen sie Y für Ja.

1.3 Local version – append to kernel release

`CONFIG_LOCALVERSION [=] [ ]`

Lokale Version – an die Kernelversion anhängen

Type: string

Hängen Sie eine zusätzliche Zeichenkette an das Ende Ihrer Kernelversion an.

Dies wird angezeigt, wenn Sie z. B. `uname` eingeben. Die hier angegebene Zeichenfolge wird an den Inhalt von einem Dateinamen mit `localversion*` als Objekt und im Quellbaum, in dieser Reihenfolge angezeigt. Die Zeichenkette darf maximal 64 Zeichen lang sein.

1.4 Automatically append version information to the version string

`CONFIG_LOCALVERSION_AUTO [=y] [Y]`

Dies versucht automatisch festzustellen, ob der aktuelle Baum ein Release-Tree ist, indem es nach **Git**-Tags sucht, die zur aktuellen Top-of-Tree-Revision gehören.

Eine Zeichenkette des Formats `-gxxxxxxx` wird der lokalen Version hinzugefügt, wenn ein git-basierter Baum gefunden wird. Die so erzeugte Zeichenkette wird nach allen passenden „`localversion*`“-Dateien und nach dem in `CONFIG_LOCALVERSION` eingestellten Wert angehängt. (Die hier tatsächlich verwendete Zeichenkette sind die ersten 12 Zeichen, die durch die Ausführung des Befehls erzeugt werden:

`$ git rev-parse --verify HEAD`
der innerhalb des Skripts „`scripts/setlocalversion`“ ausgeführt wird.)

1.5 Build ID Salt

CONFIG_BUILD_SALT [=] []

Type: string

Dies wird verwendet, um die Binaries und ihre Debug-Infos zu verknüpfen. Wenn diese Option gesetzt ist, dann wird dieser Wert in die Berechnung der Build-ID einbezogen. Wird von Distributionen verwendet, die sicherstellen wollen, dass es eindeutige IDs zwischen verschiedenen Builds gibt. Üblicherweise brauchen wir das nicht.

1.6 Kernel compression mode →

Der Linux-Kernel ist eine Art selbstextrahierende, ausführbare Datei. Es stehen mehrere Kompressionsalgorithmen zur Verfügung, die sich in Effizienz, Kompressions- und Dekompressionsgeschwindigkeit unterscheiden. Die Komprimierungsgeschwindigkeit ist nur bei der Erstellung eines Kernels relevant. Die Dekomprimierungsgeschwindigkeit ist bei jedem Systemstart von Bedeutung. (Eine ältere Version dieser Funktionalität (nur bzip2) für 2.4 wurde von Christian Ludwig bereitgestellt) Hohe Komprimierungsoptionen sind vor allem für Benutzer nützlich, die wenig Festplattenplatz zur Verfügung haben (embedded systems), für die aber die Ram-Größe weniger wichtig ist.

Überblick: Gzip werden von den älteren Kernelversionen unterstützt,

Arch Linux (since Linux/x86 5.9.0) Standard: ZSTD (former: XZ since 4.14.4, predecessor GZIP,XZ)

Debian 11.6: XZ

@TODO Weitere Linux Distros

1.6.1 Gzip

CONFIG_KERNEL_GZIP [=n] []

Die alte und bewährte gzip-Kompression. Sie bietet ein gutes Gleichgewicht zwischen Kompressionsrate und Dekompressionsgeschwindigkeit.

1.6.2 Bzip2

CONFIG_KERNEL_BZIP2 [=n] []

Die Kompressionsrate und auch die Geschwindigkeit der ist durchschnittlich. Die Geschwindigkeit der Dekomprimierung ist die langsamste. Größe des Kernels ist etwa 10 % kleiner im Vergleich zu GZIP. Es benötigt auch einen großen Speicherbereich, bei modernen Kernen benötigt man zumindest 8 MB RAM oder mehr beim Booten.

1.6.3 LZMA

CONFIG_KERNEL_LZMA [=n] []

Dieser Kompressionsalgorithmus hat die höchste Komprimierung. Die Geschwindigkeit der Dekomprimierung liegt zwischen GZIP und BZIP2. Komprimierung ist die langsamste. Kernelgröße beträgt etwa 33 % weniger als mit GZIP.

1.6.4 XZ

CONFIG_KERNEL_XZ [=n] []

XZ verwendet den LZMA2-Algorithmus und befehlssatzspezifische BCJ-Filter, die das Komprimierungsverhältnis des ausführbaren Codes verbessern können. Die Größe des Kernels ist mit XZ im Vergleich zu GZIP etwa 30 % kleiner. Auf Architekturen, für die es einen BCJ-Filter gibt (i386, x86_64, ARM, IA-64, PowerPC und SPARC), erzeugt XZ einen um einige Prozent kleineren Kernel als einfaches LZMA. Die Geschwindigkeit ist in etwa die gleiche wie bei LZMA: Die Dekomprimierungsgeschwindigkeit von XZ ist besser als die von bzip2, aber schlechter als die von gzip und LZO. Die Komprimierung ist langsam.

1.6.5 LZO

CONFIG_KERNEL_LZO [=n] []

Kompressionsrate ist die schlechteste aller anderen. Kernelgröße ist etwa 10 % größer als GZIP. Jedoch ist die Geschwindigkeit beim Komprimieren und Dekomprimieren die höchste.

1.6.6 LZ4

CONFIG_KERNEL_LZ4 [=n] []

LZ4 ist eine LZ77-Typ-Komprimierung mit einer festen, byte-orientierten Einkodierung.

Siehe auch <http://code.google.com/p/lz4>.

Komprimierungsverhältnis ist noch schlechter als LZO. 8 % größere Kernelgröße als bei LZO. Dekomprimierung ist jedoch von der Geschwindigkeit her schneller als LZO.

1.6.7 ZSTD

CONFIG_KERNEL_ZSTD [=y] [Y]

ZSTD ist ein Komprimierungsalgorithmus, der auf eine Zwischenkomprimierung mit schneller Dekomprimierungsgeschwindigkeit abzielt. Er komprimiert besser als GZIP und dekomprimiert etwa so schnell wie LZO, ist aber langsamer als LZ4. Sie benötigen mindestens 192 KB RAM oder mehr zum Booten. Das Kommandozeilenprogramm `zstd` ist für die Komprimierung erforderlich.

1.7 Default init path

CONFIG_DEFAULT_INIT [=] []

Diese Option legt den Standard-Init-Pfad für das System fest, wenn in der Kernel-Befehlszeile keine solche `init=`-Option übergeben wird. Wenn der angeforderte Pfad nicht vorhanden ist, wird trotzdem versucht, weitere Orte zu finden (z. B. `/sbin/init` usw.). Wenn dieser Pfad leer ist, wird einfach die Fallback-Liste verwendet, wenn `init=` nicht übergeben wird.

1.8 Default hostname

CONFIG_DEFAULT_HOSTNAME [=archlinux] [=archlinux]

Diese Option legt den Standard-Hostnamen des Systems fest, noch bevor der Userspace das Kommando `sethostname(2)` aufruft. Der Kernel verwendet hier traditionell `”(none)”`, Sie möchten vielleicht eine andere Voreinstellung verwenden, um ein minimales System mit weniger Konfiguration benutzbar zu machen.

1.9 System V IPC

CONFIG_SYSVIPC [=y] [Y]

Die Inter-Prozess-Kommunikation IPC ist eine Zusammenstellung aus Bibliotheksfunktionen (libraries) und Systemaufrufen die Prozesse (laufende Programme) synchronisiert und Daten untereinander austauschen kann. Generell ist das eine gute Sache, einige Programme würden auch nicht funktionieren wenn Sie hier kein Y (ja) setzen.

1.10 POSIX Message Queues

CONFIG_POSIX_MESSAGE [=y] [Y]

Die POSIX-Variante der Nachrichtenwarteschlangen (message queues) ist ein Teil der IPC. In POSIX-Nachrichtenwarteschlangen hat jede Nachricht eine Priorität, die über die Reihenfolge des Empfangs durch einen Prozess entscheidet. Wenn Sie Programme kompilieren und ausführen wollen, die z.B. für Solaris geschrieben wurden und die POSIX-Warteschlangen (Funktionen `mq_`) verwenden, sagen Sie hier Y. POSIX-Nachrichtenwarteschlangen sind via `Dateisystem` als „`mqueue`“ sichtbar und können irgendwo eingehängt werden, wenn Sie `Dateisystemoperationen` auf Nachrichtenwarteschlangen durchführen wollen.

1.11 General notification queue

CONFIG_WATCH_QUEUE [=y] [Y]

Dies ist eine allgemeine Benachrichtigungswarteschlange für den Kernel, um Ereignisse an den Userspace weiterzuleiten, indem sie in Pipes gesplittet werden. Sie kann in Verbindung mit Watches für Schlüssel-/Schlüsseländerungsbenachrichtigungen (`key/keyring`) und Gerätebenachrichtigungen verwendet werden. Bemerkung: Bei Debian Bullseye ist dies nicht gesetzt (N).

1.12 Enable process_vm_readv/writev syscalls

CONFIG.CROSS_MEMORY_ATTACH [=y] [Y]

Die Aktivierung dieser Option fügt die Systemaufrufe process_vm_readv und process_vm_writev hinzu, die es einem Prozess mit den richtigen Rechten ermöglichen, direkt aus dem Adressraum eines anderen Prozesses zu lesen oder in diesen zu schreiben. Weitere Einzelheiten finden Sie in der Manpage.

1.13 uselib syscall (for libc5 and earlier)

CONFIG.USELIB [=n] [N]

Diese Option schaltet den uselib-Systemaufruf ein, der im dynamic-Linker von libc5 und früher verwendet wird. Das aktuelle glibc verwendet diesen Systemaufruf nicht mehr, deshalb kann man diese Option ausschalten wenn sie keine Programme mehr verwenden, die auf libc5 (oder früher) compiliert wurden. Bemerkung: Debian Bullseye verwendet dies noch (Y).

1.14 Auditing support

CONFIG.AUDIT [=y] [Y]

Aktivieren Sie eine Überwachungsinfrastruktur, die mit einem anderen Kernel-Subsystem verwendet werden kann, wie z.B. SELinux (das dies für die Protokollierung der Ausgabe von avc-Nachrichten benötigt). Die Systemaufrufüberprüfung ist auf Architekturen, die sie unterstützen, enthalten.

1.15 IRQ subsystem →

Über diese Schnittstelle kann man Funktionen und Parameter für den Kernelbau auswählen. Merkmale können entweder eingebaut, modularisiert oder ignoriert werden. Parameter müssen als dezimale oder hexadezimale Zahlen oder als Text eingegeben werden.

1.15.1 Expose irq internals in debugfs

CONFIG.GENERIC_IRQ_DEBUGFS [=n] [N]

Legt interne Zustandsinformationen über debugfs offen. Hauptsächlich für Entwickler und zur Fehlersuche bei schwer zu diagnostizierenden Interrupt-Problemen.

1.16 Timers subsystem →

1.16.1 Timer tick handling →

Sie müssen aus den folgenden drei Möglichkeiten eine wählen:

1.16.1.1 Periodic timer ticks (constant rate, no dynticks)

CONFIG.HZ_PERIODIC [=n] [N]

Diese Option sorgt dafür, dass der Tick periodisch mit einer konstanten Rate läuft, auch wenn die CPU ihn nicht braucht.

1.16.1.2 Idle dynticks system (tickless idle)

CONFIG.NO_HZ_IDLE [=n] [N]

Diese Option ermöglicht ein tickloses idle-System (Leerlaufsystem): Timer-Interrupts werden nur bei Bedarf ausgelöst, wenn das System im Leerlauf ist. Dies ist v.a. zum Energiesparen interessant.

1.16.1.3 Full dynticks system (tickless)

CONFIG.NO_HZ_FULL [=y] [Y]

Diese Option ermöglicht ein tickloses idle-System (Leerlaufsystem): Timer-Interrupts werden nur bei Bedarf ausgelöst, wenn das System im Leerlauf ist. Dies ist v.a. zum Energiesparen interessant. Wird bei Linux-Distributionen ausgewählt.

1.16.2 Force user context tracking

CONFIG_CONTEXT_TRACKING_USER_FORCE [=n] [N]

Die wichtigste Voraussetzung für das Funktionieren von Full-Dynticks ist die Unterstützung des Subsystems zur Verfolgung des Benutzerkontextes. Es gibt aber auch noch andere Abhängigkeiten, die erfüllt werden müssen, damit die vollständigen Dynticks funktionieren.

Diese Option dient zum Testen, wenn eine Systemarchitektur das Backend für die Benutzerkontextverfolgung implementiert, aber noch nicht alle Anforderungen erfüllt, um die volle Dynticks-Funktion zu ermöglichen. Ohne die vollständigen Dynticks gibt es keine Möglichkeit, die Unterstützung für die Benutzerkontextverfolgung und die Teilsysteme, die darauf angewiesen sind, zu testen: RCU Userspace extended quiescent state und tickless cputime accounting. Diese Option kommt mit dem Fehlen des vollständigen dynticks-Subsystems zurecht, indem sie die Benutzerkontextverfolgung auf allen CPUs im System erzwingt.

Sagen Sie nur dann ja (Y), wenn Sie an der Entwicklung eines Architektur-Backends für die Benutzerkontextverfolgung arbeiten. Sagen Sie ansonsten N, da diese Option einen Overhead mit sich bringt, den Sie in der Praxis nicht haben wollen.

1.16.3 Old Idle dynticks config

CONFIG_NO_HZ [=y] [N]

Alte Leerlauf-Dynticks-Konfiguration

Dies ist der alte Konfigurationseintrag, der Dynticks im Leerlauf aktiviert.

~~Wir behalten ihn noch eine Weile bei, um die Abwärtskompatibilität mit älteren Konfigurationsdateien zu gewährleisten.~~

1.16.4 High Resolution Timer Support

CONFIG_HIGH_RES_TIMERS [=y] [Y]

Unterstützung von Timern mit hoher Auflösung

Diese Option aktiviert die Unterstützung hochauflösender Timer. Wenn ihre Hardware dazu nicht in der Lage ist, erhöht diese Option nur die Größe des Kernel-Images.

1.16.5 Clocksource watchdog maximum allowable skew

CONFIG_CLOCKSOURCE_WATCHDOG_MAX_SKEW_US [=100] [100]

Maximal zulässige Abweichung der Watchdog-Taktquelle

Geben Sie den maximal zulässigen Wert für den Watchdog-Versatz in Mikrosekunden an, bevor die Clocksource als instabil gemeldet wird. Der Standardwert basiert auf einem Watchdog-Intervall von einer halben Sekunde und der maximalen Frequenzdrift von NTP von 500 Teilen pro Million. Wenn die Clocksource gut genug für NTP ist, ist sie auch gut genug für den Watchdog der Clocksource!

Range: 50 – 1000

1.17 BPF subsystem →

Berkeley Packet Filter, Firewall-Filtertechnik im Kernel

1.17.1 Enable bpf() system call

CONFIG_BPF_SYSCALL [=y] [Y]

Aktivieren Sie den Systemaufruf bpf(), der es ermöglicht, BPF-Programme und -Maps über Dateideskriptoren zu manipulieren.

1.17.2 Enable BPF Just In Time compiler

CONFIG_BPF_JIT [=y] [Y]

BPF-Programme werden normalerweise von einem BPF-Interpreter verarbeitet. Diese Option ermöglicht es dem Kernel, nativen Code zu erzeugen, wenn ein Programm in den Kernel geladen wird. Dadurch wird die Verarbeitung von BPF-Programmen erheblich beschleunigt.

Beachten Sie, dass ein Administrator diese Funktion durch Ändern aktivieren sollte:

```
/proc/sys/net/core/bpf_jit_enable  
/proc/sys/net/core/bpf_jit_harden (optional)  
/proc/sys/net/core/bpf_jit_kallsyms (optional)
```

1.17.2.1 Permanently enable BPF JIT and remove BPF interpreter

CONFIG_BPF_JIT_ALWAYS_ON [=y] [Y]

Aktiviert BPF JIT und entfernt den BPF-Interpreter um spekulative Ausführungen von BPF-Anweisungen durch den Interpreter zu verhindern.

Wenn CONFIG_BPF_JIT_ALWAYS_ON eingeschaltet ist, dann wird `/proc/sys/net/core/bpf_jit_enable` permanent auf 1 gesetzt, alle Versuche diese Einstellung auf andere Werte zu legen wird mit einem Fehler zurückgewiesen.

1.17.3 Disable unprivileged BPF by default

CONFIG_BPF_UNPRIV_DEFAULT_OFF [=y] [Y]

Deaktiviert die unprivilegierte BPF standardmäßig, indem der entsprechende Eintrag

`/proc/sys/kernel/unprivileged_bpf_disabled` auf 2 gesetzt wird. Ein Administrator kann sie immer noch wieder aktivieren, indem er sie später auf 0 setzt, oder sie dauerhaft deaktiviert, indem er sie auf 1 setzt (von wo aus kein weiterer Übergang auf 0 mehr möglich ist).

Unprivilegierte BPF könnte verwendet werden, um bestimmte potenzielle Seitenkanalschwachstellen für spekulative Ausführung auf nicht gemilderter betroffener Hardware auszunutzen. Wenn Sie unsicher sind, wie Sie diese Frage beantworten sollen, antworten Sie mit Y.

1.17.4 Preload BPF file system with kernel specific program and map iterators →

BPF_PRELOAD [=n] [N]

Dadurch wird ein Kernelmodul mit mehreren eingebetteten BPF-Programmen erstellt, die als für den Menschen lesbare Dateien in den BPF-FS-Einhangepunkt eingefügt werden, was bei der Fehlersuche und der Untersuchung von BPF-Programmen und -Maps nützlich ist.

1.17.4.1 bpf_preload kernel module

Dies ist nur sichtbar wenn der übergeordnete Punkt aktiviert ist.

CONFIG_BPF_PRELOAD_UMD [=m] []

Dadurch wird ein Kernelmodul mit mehreren eingebetteten BPF-Programmen erstellt, die als für den Menschen lesbare Dateien in den BPF-FS-Einhangepunkt eingefügt werden, was bei der Fehlersuche und der Untersuchung von BPF-Programmen und -Maps nützlich ist.

1.17.5 Enable BPF LSM Instrumentation

CONFIG_BPF_LSM [=y] [Y]

Ermöglicht die Instrumentierung der Sicherheitshaken mit BPF-Programmen zur Implementierung dynamischer MAC- und Prüfungsrichtlinien. Wenn Sie unsicher sind, wie Sie diese Frage beantworten sollten, antworten Sie mit N.

1.18 Preemption Model (Preemptible Kernel (Low-Latency Desktop)) →

Eingestellt auf : Low-Latency, d.h. nur kleine Verzögerungen beim Modell des Multitaskings. Es gibt drei Einstellungen:

1.18.1 No Forced Preemption (Server)

CONFIG_PREEMPT_NONE [=n] [N]

Das war das traditionelle Linux Modell der Unterbrechungen, das sich auf den Durchsatz konzentrierte. Wird vor allem für den Server-Einsatz verwendet. Es gibt durchaus gute Performance für die Latenz, jedoch keine Garantie dafür und es kann zu zufälligen, längeren Verzögerungszeiten kommen.

Für einen Serverbetrieb wird diese Einstellung empfohlen, damit der maximale Durchsatz an Rechenleistung entsteht.

1.18.2 Voluntary Kernel Preemption (Desktop)

CONFIG_PREEMPT_VOLUNTARY [=n] [N]

Diese Einstellung reduziert die Latenz des Kernels durch zusätzliche „explizite Unterbrechungspunkte“, im Kernel. Diese neuen Unterbrechungspunkte wurden ausgewählt, um die maximale Latenz beim neuerlichen Zuordnen des Schedulers zu reduzieren und dadurch schnelle Reaktionszeiten der Applikationen zu gewährleisten. – Auf Kosten eines geringeren Durchsatzes wird dies erreicht.

1.18.3 Preemptible Kernel (Low-Latency Desktop)

CONFIG_PREEMPT [=y] [Y]

Bei dieser Einstellung wird die Latenz des Kernels weiter erniedrigt indem der gesamte Code des Kernels (keine kritischen, geschützten Bereiche) unterbrechbar gemacht wird. Dadurch wird ein reibungsloses Arbeiten mit Applikationen aus Nutzersicht erreicht, sogar unter Volllast. Wähle diese Einstellung, wenn man einen Desktop oder ein Embedded-System mit einer Latenz im Millisekundenbereich möchte. Natürlich geht diese Einstellung mit einem leicht geringeren Durchsatz an Rechenleistung einher.

1.19 Preemption behaviour defined on boot

CONFIG_PREEMPT_DYNAMIC [=y] [Y]

Diese Option ermöglicht es, das Präemptionsmodell über den Kernel-Kommandozeilenparameter zu definieren und damit das während der Kompilierung definierte Standard-Präemptionsmodell außer Kraft zu setzen. Diese Funktion ist vor allem für Linux-Distributionen interessant, die eine vorgefertigte Kernel-Binärdatei bereitstellen, um die Anzahl der angebotenen Kernel-Varianten zu reduzieren und dennoch verschiedene Anwendungsfälle zu ermöglichen.

Der Laufzeit-Overhead ist vernachlässigbar, wenn HAVE_STATIC_CALL_INLINE aktiviert ist, aber wenn Laufzeit-Patching für die spezifische Architektur nicht verfügbar ist, sollte der potenzielle Overhead in Betracht gezogen werden. Interessant wird es, wenn derselbe vorgefertigte Kernel sowohl für Servers als auch für Desktop-Workloads verwendet werden soll.

1.20 Core Scheduling for SMT

CONFIG_SCHED_CORE [=y] [Y]

Kern-Scheduling für SMT

Diese Option ermöglicht Core Scheduling, ein Mittel zur koordinierten Auswahl von Aufgaben zwischen SMT-Geschwistern. Wenn diese Option aktiviert ist - siehe prctl (PR_SCHED_CORE) - stellt die Aufgabenauswahl sicher, dass alle SMT-Geschwister eine Aufgabe aus der gleichen „Kerngruppe“ ausführen und den Leerlauf erzwingen, wenn keine passende Aufgabe gefunden wird. Diese Funktion wird unter anderem verwendet:

- Entschärfung einiger (nicht aller) SMT-Seitenkanäle;
- Begrenzung der SMT-Interferenz zur Verbesserung des Determinismus und/oder der Leistung.

SCHED_CORE ist standardmäßig deaktiviert. Wenn es aktiviert und unbenutzt ist, was bei Linux-Distributionen wahrscheinlich der Fall ist, sollte es keine messbaren Auswirkungen auf die Leistung haben.

1.21 CPU/Task time and stats accounting →

1.21.1 Cputime accounting (Full dynticks CPU time accounting) →

1.21.1.1 Full dynticks CPU time accounting

CONFIG_VIRT_CPU_ACCOUNTING_GEN [=y] [Y]

Wählen Sie diese Option, um die Berechnung der Task- und CPU-Zeit auf Full-Dynticks-Systemen zu aktivieren. Diese Berechnung wird durch die Überwachung aller Kernel-Benutzer-Grenzen mithilfe des Kontextverfolgungs-Subsystems implementiert.

Die Berechnung erfolgt daher auf Kosten eines erheblichen Overheads.

Im Moment ist dies nur sinnvoll, wenn Sie an der Entwicklung des vollständigen Dynticks-Subsystems arbeiten.

1.21.2 Fine granularity task level IRQ time accounting

CONFIG_IRQ_TIME_ACCOUNTING [=y] [Y]

Wählen Sie diese Option aus, um eine fein granulare Berechnung der Task-Irq-Zeit zu aktivieren. Dies geschieht durch das Lesen eines Zeitstempels bei jedem Übergang zwischen dem softirq- und dem hardirq-Zustand, so dass es zu geringen Leistungseinbußen kommen kann.

Im Zweifelsfall sagen Sie hier N für Nein.

1.21.3 BSD Process Accounting

CONFIG_BSD_PROCESS_ACCT [=y] [Y]

Wenn Sie hier Y (für Ja) angeben, kann ein Programm auf Benutzerebene den Kernel (über einen speziellen Systemaufruf) anweisen, Prozessabrechnungsinformationen in eine Datei zu schreiben: Jedes Mal, wenn ein Prozess beendet wird, werden Informationen über diesen Prozess vom Kernel an die Datei angehängt. Die Informationen beinhalten Dinge wie die Erstellungszeit, den besitzenden Benutzer, den Befehlsnamen, den Speicherverbrauch, das kontrollierende Terminal usw. (die vollständige Liste kann in der acct-Struktur in <file:include/linux/acct.h> gefunden werden). Es obliegt dem Programm auf Benutzerebene, nützliche Dinge mit diesen Informationen zu tun. Dies ist im Allgemeinen eine gute Idee, also sagen Sie Y für Ja.

1.21.3.1 BSD Process Accounting version 3 file format

CONFIG_BSD_PROCESS_ACCT_V3 [=y] [Y]

Wenn Sie hier Y (für Ja) angeben, werden die Prozessabrechnungsinformationen in ein neues Dateiformat geschrieben, das auch die Prozess-IDs der einzelnen Prozesse und ihrer Eltern protokolliert. Beachten Sie, dass dieses Dateiformat nicht mit den früheren v0/v1/v2-Dateiformaten kompatibel ist, so dass Sie aktualisierte Werkzeuge für die Verarbeitung benötigen. Eine vorläufige Version dieser Werkzeuge ist unter <<http://www.gnu.org/software/acct/>> verfügbar.

1.21.4 Export task/process statistics through netlink

CONFIG_TASKSTATS [=y] [Y]

Export ausgewählter Statistiken für Aufgaben/Prozesse über die generische Netlink-Schnittstelle. Im Gegensatz zur BSD-Prozessabrechnung sind die Statistiken während der Lebensdauer von Aufgaben/Prozessen als Antwort auf Befehle verfügbar. Wie BSD-Accounting werden sie beim Beenden von Tasks in den Benutzerbereich gesendet.

Sagen Sie N, wenn Sie unsicher sind.

1.21.4.1 Enable per-task delay accounting

CONFIG_TASK_DELAY_ACCT [=y] [Y]

Sammeln Sie Informationen über die Zeit, die eine Task für das Warten auf Systemressourcen wie CPU, synchrone Block-E/A-Abwicklung und Auslagerung von Seiten aufwendet. Solche Statistiken können bei der Festlegung der Prioritäten eines Tasks im Verhältnis zu anderen Tasks für CPU-, IO-, RSS-Limits usw. helfen.

Sagen Sie N, wenn Sie unsicher sind.

1.21.4.2 Enable extended accounting over taskstats

CONFIG_TASK_XACCT [=y] [Y]

Sammeln von erweiterten Task-Accounting-Daten und Senden der Daten an das Userland zur Verarbeitung über die Taskstats-Schnittstelle.

Sagen Sie N, wenn Sie unsicher sind.

1.21.4.2.1 Enable per-task storage I/O accounting

CONFIG_TASK_IO_ACCOUNTING [=y] [Y]

Sammeln von Informationen über die Anzahl der Bytes an Speicher-E/A, die dieser Task verursacht hat.

Sagen Sie N, wenn Sie unsicher sind.

1.21.5 Pressure stall information tracking

CONFIG_PSI [=y] [Y]

Sammeln Sie Metriken, die anzeigen, wie überlastet die CPU-, Speicher- und IO-Kapazität im System sind.

Wenn Sie hier Y angeben, erstellt der Kernel `/proc/pressure/` mit die Druckstatistikdateien `cpu`, `memory` und `io`. Diese zeigen den Anteil der Wailtime an, in dem einige oder alle Tasks im System aufgrund der Beanspruchung der jeweiligen Ressource verzögert sind.

In Kernen mit cgroup-Unterstützung verfügen cgroups (nur `cgroup2`) über `cpu.pressure`-, `memory.pressure`- und `io.pressure`-Dateien, die nur die Druckstaus für die gruppierten Aufgaben zusammenfassen.

Weitere Einzelheiten finden Sie unter `Documentation/accounting/psi.rst`.

Sagen Sie N, wenn Sie unsicher sind.

1.21.5.1 Require boot parameter to enable pressure stall information tracking

CONFIG_PSI_DEFAULT_DISABLED [=n] [N]

Wenn diese Option gesetzt ist, ist die Verfolgung von Druckstauinformationen standardmäßig deaktiviert, kann aber durch die Übergabe von `psi=1` auf der Kernel-Befehlszeile beim Booten aktiviert werden.

Diese Funktion fügt dem Task-Wakeup- und Sleep-Pfad des Schedulers etwas Code hinzu. Der Overhead ist zu gering, um gängige planungsintensive Arbeitslasten in der Praxis zu beeinträchtigen (z. B. Webserver, Memcache), aber es zeigt sich in künstlichen Scheduler-Stresstests, wie z. B. `Hackbench`. Wenn Sie paranoid sind und nicht sicher, wofür der Kernel verwendet wird, sagen Sie Y für Ja.

Sagen Sie N, wenn Sie unsicher sind.

1.22 CPU isolation

CONFIG_CPU_ISOLATION [=y] [Y]

Stellen Sie sicher, dass CPUs, auf denen kritische Aufgaben laufen, nicht durch irgendwelche „Störquellen“ wie ungebundene Workqueues, Timers, kthreads usw. gestört werden.

Ungebundene Aufgaben werden auf Housekeeping-CPU's verlagert. Dies wird durch den Boot-Parameter „`isolcpus`“ gesteuert.

Sagen Sie Y für ja, wenn Sie unsicher sind.

1.23 RCU Subsystem →

Read – Copy – Update (Lesen, Kopieren, Aktualisieren)

1.23.1 Make expert-level adjustments to RCU configuration

CONFIG_RCU_EXPERT [=y] [Y]

Diese Option muss aktiviert werden, wenn Sie Anpassungen der RCU-Konfiguration auf Expertenebene vornehmen möchten. Standardmäßig können solche Anpassungen nicht vorgenommen werden, was den oft vorteilhaften Nebeneffekt hat, dass „`make oldconfig`“ Sie davon abhält, alle möglichen detaillierten Fragen darüber zu stellen, wie Sie zahlreiche obskure RCU-Optionen eingerichtet haben möchten.

Sagen Sie Y, wenn Sie Anpassungen an RCU auf Expertenebene vornehmen müssen.

Sagen Sie N, wenn Sie unsicher sind.

1.23.2 Force selection of TASKS_RCU

CONFIG_FORCE_TASKS_RCU [=n] [N]

Diese Option erzwingt eine aufgabenbasierte RCU-Implementierung die nur freiwillige Kontextwechsel verwendet (keine Preemption!), Leerlauf und Benutzermodus-Ausführung als Ruhezustände verwendet. Nicht für manuelle Auswahl in den meisten Fällen.

1.23.3 Force selection of Tasks Rude RCU

CONFIG_FORCE_TASKS_RUDE_RCU [=n] [N]

Diese Option erzwingt eine Task-basierte RCU-Implementierung, die nur Kontextwechsel (einschließlich Preemption) und die Ausführung im Benutzermodus als Ruhezustand verwendet. Sie erzwingt IPIs und

Kontextwechsel auf allen Online-CPUs, auch auf den Idle-CPU, also mit Vorsicht verwenden. In den meisten Fällen nicht für die manuelle Auswahl geeignet.

1.23.4 Force selection of Tasks Trace RCU

CONFIG_FORCE_TASKS_TRACE_RCU [=n] [N]

Diese Option ermöglicht eine Task-basierte RCU-Implementierung, die explizite `rcu_read_lock_trace()`-Lesemarker verwendet und es ermöglicht, dass diese Leser sowohl in der Leerlaufschleife als auch in den CPU-Hotplug-Codepfaden erscheinen. Es kann IPIs auf Online-CPU erzwingen, auch auf Idle-CPU, also mit Vorsicht verwenden. In den meisten Fällen nicht für die manuelle Auswahl geeignet.

1.23.5 Tree-based hierarchical RCU fanout value

CONFIG_RCU_FANOUT [=64] [64]

Diese Option steuert den Fanout von hierarchischen Implementierungen von RCU, so dass RCU auf Maschinen mit einer großen Anzahl von CPUs effizient arbeiten kann. Dieser Wert muss mindestens die vierte Wurzel von `NR_CPUS` sein, wodurch `NR_CPUS` wahnsinnig groß werden kann. Der Standardwert von `RCU_FANOUT` sollte für Produktionssysteme verwendet werden, aber wenn Sie die RCU-Implementierung selbst einem Stresstest unterziehen, ermöglichen kleine `RCU_FANOUT`-Werte das Testen von Codepfaden für große Systeme auf kleinen (kleineren) Systemen.

Wählen Sie eine bestimmte Zahl, wenn Sie RCU selbst testen. Nehmen Sie den Standardwert, wenn Sie unsicher sind.

Symbol: `RCU_FANOUT` [=64]

Type : integer (Ganzzahl)

Range : [2 64]

1.23.6 Tree-based hierarchical RCU leaf-level fanout value

CONFIG_RCU_FANOUT_LEAF [=16] [16]

Diese Option steuert das Fanout auf Blattebene bei hierarchischen Implementierungen von RCU und ermöglicht es, Cache-Misses gegen Sperrkonflikte abzuwägen. Systeme, die ihre Scheduling-Clock-Interrupts aus Gründen der Energieeffizienz synchronisieren, werden die Standardeinstellung bevorzugen, da der kleinere Leaf-Level-Fanout die Lock-Contention-Level akzeptabel niedrig hält. Sehr große Systeme (Hunderte oder Tausende von CPUs) werden stattdessen diesen Wert auf den maximal möglichen Wert setzen wollen, um die Anzahl der Cache-Misses zu reduzieren, die während der Initialisierung der RCU-Grace-Periode auftreten. Diese Systeme neigen dazu, CPU-gebunden zu laufen, und werden daher nicht von synchronisierten Interrupts unterstützt, und neigen daher dazu, sie zu verzerren, was den Sperrkonflikt so weit reduziert, dass große Fanouts auf Blattebene gut funktionieren. Das heißt, wenn Sie den Fanout auf Blattebene auf eine große Zahl setzen, wird dies wahrscheinlich zu problematischen Sperrkonflikten auf den `rcu_node`-Strukturen auf Blattebene führen, es sei denn, Sie booten mit dem Kernelparameter `skew_tick`.

Wählen Sie eine bestimmte Zahl, wenn Sie die RCU selbst testen.

Wählen Sie den maximal zulässigen Wert für große Systeme, aber bedenken Sie, dass Sie möglicherweise auch den Kernel-Boot-Parameter `skew_tick` setzen müssen, um Konflikte bei den Sperren der `rcu_node`-Strukturen zu vermeiden. Nehmen Sie den Standardwert, wenn Sie unsicher sind.

Symbol: `RCU_FANOUT_LEAF` [=64]

Type : integer (Ganzzahl)

Range : [2 64]

1.23.7 Enable RCU priority boosting

CONFIG_RCU_BOOST [=y] [Y]

Diese Option erhöht die Priorität von preemptierten RCU-Lesern, die die aktuelle preemptible RCU-Schonfrist zu lange blockieren. Diese Option verhindert auch, dass schwere Lasten den Aufruf von RCU-Callbacks blockieren.

Geben Sie hier Y an, wenn Sie mit Echtzeitanwendungen oder großen Lasten arbeiten.

Sagen Sie hier N ein, wenn Sie unsicher sind.

1.23.7.1 Milliseconds to delay boosting after RCU grace-period start

CONFIG.RCU_BOOST_DELAY [=500] [500]

Diese Option gibt die Zeit an, die nach dem Beginn einer bestimmten Karenzzeit gewartet werden soll, bevor die Priorität von RCU-Lesern, die diese Karenzzeit blockieren, erhöht wird.

Beachten Sie, dass jeder RCU-Leser, der eine beschleunigte RCU-Schonfrist blockiert, sofort hochgestuft wird.

Akzeptieren Sie die Standardeinstellung, wenn Sie unsicher sind.

Symbol: RCU_BOOST_DELAY [=500]

Typ : Integer (Ganzzahl)

Bereich : [0 3000]

1.23.7.2 Perform RCU expedited work in a real-time kthread

CONFIG.RCU_EXP_KTHREAD [=n] [N]

Verwenden Sie diese Option, um die Latenzzeiten der beschleunigten Neuheitsschonfristen weiter zu reduzieren, was allerdings mit mehr Störungen verbunden ist. Diese Option ist standardmäßig auf PREEMPT_RT=y-Kerneln deaktiviert, die beschleunigte Neuheitsschonfristen nach dem Booten durch die bedingungslose Einstellung rcupdate.rcu_normal_after_boot=1 deaktivieren.

Akzeptieren Sie die Voreinstellung, wenn Sie unsicher sind.

1.23.8 Offload RCU callback processing from boot-selected CPUs

CONFIG.RCU_NOCB_CPU [=y] [Y]

Verwenden Sie diese Option, um den Jitter des Betriebssystems für aggressive HPC- oder Echtzeit-Workloads zu reduzieren. Sie kann auch verwendet werden, um RCU-Callback-Aufrufe auf energieeffiziente CPUs in batteriebetriebenen asymmetrischen Multiprozessoren auszulagern. Der Preis für diesen reduzierten Jitter ist, dass der Overhead von call_rcu() ansteigt und dass bei einigen Workloads ein erheblicher Anstieg der Kontextwechselraten zu verzeichnen ist.

Diese Option entlastet den Aufruf von Callbacks von der Gruppe von CPUs, die zur Boot-Zeit durch den rcu_nocbs-Parameter angegeben wird. Für jede dieser CPUs wird ein kthread („rcuox/N“) erstellt, um Callbacks aufzurufen, wobei „N“ die CPU ist, die entlastet wird, und wobei „x“ „p“ für RCU-preempt (PREEMPTION-Kernel) und „s“ für RCU-sched (!PREEMPTION-Kernel) ist. Nichts hindert diesen kthread daran, auf den angegebenen CPUs zu laufen, aber (1) die kthreads können zwischen jedem Callback preempted werden, und (2) Affinität oder cgroups können verwendet werden, um die kthreads zu zwingen, auf jeder gewünschten Gruppe von CPUs zu laufen.

Sagen Sie hier Y, wenn Sie trotz des zusätzlichen Overheads ein geringeres OS-Jitter benötigen.

Sagen Sie hier N, wenn Sie unsicher sind.

1.23.8.1 Offload RCU callback processing from all CPUs by default

CONFIG.RCU_NOCB_CPU_DEFAULT_ALL [=n] [N]

Verwenden Sie diese Option, um die Callback-Verarbeitung standardmäßig von allen CPUs zu entlasten, wenn der Boot-Parameter rcu_nocbs oder nohz_full nicht vorhanden ist. Dadurch wird auch die Notwendigkeit vermieden, Boot-Parameter zu verwenden, um den Effekt der Entlastung aller CPUs beim Booten zu erreichen.

Geben Sie hier Y an, wenn Sie alle CPUs standardmäßig beim Booten entlasten wollen.

Sagen Sie hier N, wenn Sie sich nicht sicher sind.

1.23.8.2 Offload RCU callback from real-time kthread

CONFIG.RCU_NOCB_CPU_CB_BOOST [=n] [N]

Verwenden Sie diese Option, um ausgelagerte Rückrufe als SCHED_FIFO aufzurufen, um ein Aushungern durch schwere SCHED_OTHER-Hintergrundlast zu vermeiden. Natürlich führt die Ausführung als SCHED_FIFO während Callback Floods dazu, dass die rcuo[ps] kthreads die CPU für Hunderte von Millisekunden oder mehr monopolisieren. Wenn Sie diese Option aktivieren, müssen Sie daher sicherstellen, dass latenzempfindliche Aufgaben entweder mit höherer Priorität oder auf einer anderen CPU ausgeführt werden.

Geben Sie hier Y an, wenn Sie die RT-Priorität für die Auslagerung von kthreads festlegen möchten.

Sagen Sie hier N, wenn Sie einen !PREEMPT_RT-Kernel bauen und sich unsicher sind.

1.23.9 Tasks Trace RCU readers use memory barriers in user and idle

CONFIG_TASKS_TRACE_RCU_READ_MB [=n] [N]

Verwenden Sie diese Option, um die Anzahl der IPIs (inter-processor interrupts), die an CPUs gesendet werden, die im Benutzerraum ausgeführt werden oder sich im Leerlauf befinden, während Tasks RCU-Tilgungsfristen verfolgen, weiter zu reduzieren. Da eine vernünftige Einstellung des Kernel-Boot-Parameters rcupdate.rcu_task_ipi_delay solche IPIs für viele Arbeitslasten eliminiert, ist die richtige Einstellung dieser Kconfig-Option vor allem für aggressive Echtzeiteinstellungen und für batteriebetriebene Geräte wichtig, daher die oben gewählte Standardeinstellung.

Sagen Sie hier Y, wenn Sie IPIs hassen.

Sagen Sie hier N, wenn Sie leseseitige Speicherbarrieren hassen.

Nehmen Sie die Standardeinstellung, wenn Sie unsicher sind.

1.23.10 RCU callback lazy invocation functionality

CONFIG_RCU_LAZY [=y] [Y]

Um Strom zu sparen, sollten Sie RCU-Rückrufe stapeln und nach einer Verzögerung, einem Speicherdruck oder einer zu großen Rückrufliste flushen.

1.23.11 RCU callback-batch backup time check

CONFIG_RCU_DOUBLE_CHECK_CB_TIME [=y] [Y]

Verwenden Sie diese Option, um eine präzisere Durchsetzung des Modulparameters rcutree.rcu_resched_ns in Situationen zu ermöglichen, in denen ein einziger RCU-Callback Hunderte von Mikrosekunden lang laufen könnte, wodurch die 32-Callback-Batching-Funktion, die verwendet wird, um die Kosten der feinkörnigen, aber teuren local_clock()-Funktion zu amortisieren, unterlaufen wird.

Diese Option rundet rcutree.rcu_resched_ns auf den nächsten Jiffy auf und setzt die 32-Callback-Batching-Funktion außer Kraft, wenn diese Grenze überschritten wird.

Sagen Sie hier Y, wenn Sie eine strengere Durchsetzung des Rückruffimits benötigen.

Sagen Sie hier N, wenn Sie unsicher sind.

1.24 Kernel .config support

CONFIG_IKCONFIG [=y] [Y]

Mit dieser Option kann der gesamte Inhalt der „.config“-Datei des Linux-Kernels im Kernel gespeichert werden. Sie dokumentiert, welche Kernel-Optionen in einem laufenden Kernel oder in einem On-Disk-Kernel verwendet werden. Diese Informationen können mit dem Skript scripts/extract-ikconfig aus der Kernel-Image-Datei extrahiert und als Eingabe verwendet werden, um den aktuellen Kernel neu zu erstellen oder einen anderen Kernel zu bauen. Sie können auch aus einem laufenden Kernel extrahiert werden, indem /proc/config.gz gelesen wird, falls dies aktiviert ist (siehe unten).

Definiert mit init/Kconfig:686

1.24.1 Enable access to .config through /proc/config.gz

CONFIG_IKCONFIG_PROC [=y] [Y]

Diese Option ermöglicht den Zugriff auf die Kernelkonfigurationsdatei über /proc/config.gz.

1.25 Enable kernel headers through /sys/kernel/kheaders.tar.xz

CONFIG_IKHEADERS [=m] [M]

Diese Option ermöglicht den Zugriff auf die In-Kernel-Header, die während des Build-Prozesses erzeugt werden. Diese können verwendet werden, um eBPF-Tracing-Programme oder ähnliche Programme zu erstellen. Wenn Sie die Header als Modul erstellen, wird ein Modul namens kheaders.ko erstellt, das bei Bedarf geladen werden kann, um Zugriff auf die Header zu erhalten.

1.26 Kernel log buffer size (16 ⇒ 64KB, 17 ⇒ 128KB)

CONFIG_LOG_BUF_SHIFT [=17] [17]

Wählen Sie die minimale Größe des Kernel-Protokollpuffers als eine Potenz von 2 aus. Die endgültige Größe wird durch den Konfigurationsparameter LOG_CPU_MAX_BUF_SHIFT beeinflusst, siehe unten.

Eine höhere Größe kann auch durch den Boot-Parameter „log_buf_len“ erzwungen werden.

Beispiele:

17 $\Rightarrow$ 128 KB

16 $\Rightarrow$ 64 KB

15 $\Rightarrow$ 32 KB

14 $\Rightarrow$ 16 KB

13 $\Rightarrow$ 8 KB

12 $\Rightarrow$ 4 KB

Symbol: LOG_BUF_SHIFT

Type: Integer (Ganzzahl)

Range: [12 25]

1.27 CPU kernel log buffer size contribution (13 $\Rightarrow$ 8 KB, 17 $\Rightarrow$ 128KB)

CONFIG_LOG_BUF_SHIFT [=12] [12]

Diese Option ermöglicht es, die Standardgröße des Ringpuffers entsprechend der Anzahl der CPUs zu erhöhen. Der Wert definiert den Beitrag jeder CPU als eine Potenz von 2. Der beanspruchte Speicherplatz beträgt in der Regel nur wenige Zeilen, kann aber viel mehr sein, wenn Probleme gemeldet werden, z. B. bei Rückverfolgungen. Die erhöhte Größe bedeutet, dass ein neuer Puffer zugewiesen werden muss und der ursprüngliche statische Puffer ungenutzt ist. Dies ist nur auf Systemen mit mehr CPUs sinnvoll. Daher wird dieser Wert nur verwendet, wenn die Summe der Beiträge größer ist als die Hälfte des Standard-Kernel-Ringpuffers, wie durch LOG_BUF_SHIFT definiert. Die Standardwerte sind so eingestellt, dass mehr als 16 CPUs erforderlich sind, um die Zuweisung auszulösen. Diese Option wird auch ignoriert, wenn der Kernelparameter „log_buf_len“ verwendet wird, da er eine exakte (Zweierpotenz) Größe des Ringpuffers erzwingt. Die Anzahl der möglichen CPUs wird für diese Berechnung verwendet, wobei Hot-plugging ignoriert wird, so dass die Berechnung für das Worst-Case-Szenario optimal ist und gleichzeitig ein einfacher Algorithmus ab dem Hochfahren verwendet werden kann. Beispiele für Verschiebungswerte und ihre Bedeutung: 17 $\Rightarrow$ 128 KB für jede CPU

16 $\Rightarrow$ 64 KB für jede CPU

15 $\Rightarrow$ 32 KB für jede CPU

14 $\Rightarrow$ 16 KB für jede CPU

13 $\Rightarrow$ 8 KB für jede CPU

12 $\Rightarrow$ 4 KB für jede CPU

Symbol: LOG_CPU_MAX_BUF_SHIFT

Type: Integer (Ganzzahl)

Range: [0 21]

1.28 Printk indexing debugfs interface)

CONFIG_PRINTK_INDEX [=y] [Y]

Unterstützung für die Indizierung aller zur Kompilierzeit bekannten printk-Formate unter <debugfs>/printk/index/<module> hinzufügen. Dies kann als Teil der Wartung von Daemonen, die /dev/kmsg überwachen, verwendet werden, da es die Überprüfung der in einem Kernel vorhandenen printk-Formate erlaubt, was die Erkennung von Fällen ermöglicht, in denen überwachte printks geändert oder nicht mehr vorhanden sind.

Es gibt keine zusätzlichen Laufzeitkosten für printk, wenn dies aktiviert ist.

1.29 Scheduler features $\rightarrow$

Scheduler-Funktionen

1.29.1 Enable utilization clamping for RT/FAIR tasks

CONFIG_UCLAMP_TASK [=y] [Y]

Diese Funktion ermöglicht es dem Scheduler, die geklemmte Auslastung jeder CPU auf der Grundlage der auf dieser CPU geplanten RUNNABLE-Tasks zu verfolgen. Mit dieser Option kann der Benutzer die minimale und maximale CPU-Auslastung angeben, die für RUNNABLE-Aufgaben zulässig ist. Die maximale Auslastung definiert die maximale Häufigkeit, mit der ein Task laufen soll, während die minimale Auslastung die minimale Häufigkeit definiert, mit der er laufen soll.

Sowohl die Minimal- als auch die Maximalwerte für die Auslastung sind Hinweise für den Scheduler, um seine Frequenzauswahl zu verbessern, aber sie erzwingen oder gewähren keine bestimmte Bandbreite für Tasks.

Im Zweifelsfall sagen Sie N für Nein.

1.29.1.1 Number of supported utilization clamp buckets

`CONFIG.UCLAMP.BUCKETS.COUNT [=5] [5]`

Legt die Anzahl der zu verwendenden Klammerbereiche fest. Der Bereich der einzelnen Buckets ist `SCHED.CAPACITY_SCALE/UCLAMP.BUCKETS.COUNT`. Je höher die Anzahl der Clamp-Buckets, desto feiner die Granularität und desto höher die Präzision der Clamp-Aggregation und -Verfolgung während der Laufzeit. Mit dem minimalen Konfigurationswert haben wir beispielsweise 5 Clamp-Buckets, die jeweils 20 % Auslastung verfolgen. Eine um 25 % gesteigerte Aufgabe wird im Bucket [20..39]% gezählt und setzt den effektiven Wert der Bucketklemme auf 25 %. Wenn eine zweite, um 30 % erhöhte Aufgabe auf derselben CPU eingeplant wird, wird diese Aufgabe im selben Bucket wie die erste Aufgabe gezählt und erhöht den effektiven Bucket-Clamp-Wert auf 30 %. Der effektive Klemmwert eines Bereichs wird auf seinen Nennwert (20 % im obigen Beispiel) zurückgesetzt, wenn keine weiteren Aufgaben mehr in diesem Bereich gezählt werden. Bei einigen Aufgaben kann eine zusätzliche Verstärkungs-/Kappungsmarge hinzugefügt werden. Im obigen Beispiel wird die 25 %-Aufgabe auf 30 % angehoben, bis sie die CPU verlässt. Sollte dies auf bestimmten Systemen nicht akzeptabel sein, ist es immer möglich, den Spielraum zu verringern, indem die Anzahl der Clamp-Buckets erhöht wird, um den verbrauchten Speicher gegen die Genauigkeit der Laufzeitverfolgung einzutauschen.

Im Zweifelsfall sollten Sie den Standardwert verwenden.

1.30 Memory placement aware NUMA scheduler

`CONFIG.NUMA.BALANCING [=y] [Y]`

Diese Option fügt Unterstützung für automatische NUMA-bewusste Speicher-/Task-Platzierung hinzu. Der Mechanismus ist recht primitiv und basiert darauf, dass Speicher migriert wird, wenn er Referenzen auf den Knoten hat, auf dem die Aufgabe läuft.

Dieses System ist auf UMA-Systemen inaktiv.