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 Standarwert ein Nein [n], meine persönliche Einstellung ein Ja [Y].

Wenn Sie Tippfehler finden oder Korrekturen wünschen, dann schicken Sie dies mit Erläuterungen und dem Hinweis auf die obenstehende Version V6.6 an: oe1tkt@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 `localverion*` 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 `-gxxxxxxxx` 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 eineindeutige 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 Kerneln 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 Dekomrimieren die höchste.

1.6.6 LZ4

CONFIG_KERNEL_LZ4 [=n] [ ]
LZ4 ist eine LZ77-Typ-Komprimierung mit einer festen, byte-orientierten Enkodierung.
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_MQUEUE [=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 Abwärtskompatiblität 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-Einhängepunkt 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-Einhängepunkt 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 geringerem Durchsatz an Rechenleistung einher.

1.19 Preemtion 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 Server- 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 Walltime an, in dem einige oder alle Tasks im System aufgrund der Beanspruchung der jeweiligen Ressource verzögert sind.

In Kerneln 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-CPUs 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-CPUs, 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-CPUs erzwingen, auch auf Idle-CPUs, 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 Echtzeitinstallationen 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 Rückruflimits 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 ⇒ 128 KB
- 16 ⇒ 64 KB
- 15 ⇒ 32 KB
- 14 ⇒ 16 KB
- 13 ⇒ 8 KB
- 12 ⇒ 4 KB

Symbol: LOG_BUF_SHIFT
Type: Integer (Ganzzahl)
Range: [12 25]

1.27 CPU kernel log buffer size contribution (13 ⇒ 8 KB, 17 ⇒ 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 Hotplugging 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 ⇒ 128 KB für jede CPU
16 ⇒ 64 KB für jede CPU
15 ⇒ 32 KB für jede CPU
14 ⇒ 16 KB für jede CPU
13 ⇒ 8 KB für jede CPU
12 ⇒ 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 →

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 bietet Unterstützung für die automatische NUMA-kompatible Speicher-/Task-Platzierung. 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.

1.30.1 Automatically enable NUMA aware memory/task placemnent

CONFIG_NUMA_BALANCING_DEFAULT_ENABLED [=y] **[Y]**
Wenn diese Option gesetzt ist, wird der automatische NUMA-Ausgleich aktiviert, wenn das System auf einem NUMA-Rechner läuft.

1.31 Control Group support →

CONFIG_CGROUPS [=y] **[Y]**
(Unterstützung der Kontrollgruppe)
Diese Option bietet Unterstützung für die Gruppierung von Prozessgruppen zur Verwendung mit Prozesskontrollsubsystemen wie Cpusets, CFS, Speicherkontrolle oder Geräteisolierung.
Siehe

- Dokumentation/scheduler/sched-design-CFS.rst (CFS)
- Documentation/admin-guide/cgroup-v1/ (Funktionen für Gruppierung, Isolierung und Ressourcenkontrolle)

Sagen Sie N, wenn Sie unsicher sind.

1.31.1 Favor dynamic modification latency reduction by default

CONFIG_CGROUP_FAVOR_DYNMODS [=n] **[N]**
Diese Option aktiviert standardmäßig die Einhängeoption „favordynmods“, die die Latenzzeiten dynamischer C-Gruppen-Änderungen wie Task-Migrationen und Controller-Ein-/Ausschaltungen auf Kosten von Hot-Path-Operationen wie Forks und Exits verteuert.
Sagen Sie N, wenn Sie unsicher sind.

1.31.2 Memory controller

CONFIG_MEMCG [=y] **[Y]**
Ermöglicht die Kontrolle über den Speicherbedarf von Tasks in einer cgroup.

1.31.3 IO controller

CONFIG_BLK_CGROUP [=y] **[Y]**
Generische Block IO Controller cgroup Schnittstelle. Dies ist die gemeinsame cgroup-Schnittstelle, die von verschiedenen IO-Kontrollstrategien verwendet werden sollte.
Derzeit wird sie vom CFQ IO Scheduler zur Erkennung von Task-Gruppen und zur Steuerung der Zuweisung von Festplattenbandbreite (proportionale Zeitscheibenzuweisung) an solche Task-Gruppen verwendet. Sie wird auch von der Bio-Throttling-Logik in der Blockschicht verwendet, um eine Obergrenze

für die IO-Raten auf einem Gerät einzuführen.
Diese Option aktiviert nur die generische Infrastruktur des Block-IO-Controllers. Man muss auch die tatsächliche IO-Kontrolllogik/-Politik aktivieren. Um die proportionale Aufteilung der Festplattenbandbreite in CFQ zu aktivieren, setzen Sie CONFIG_BFQ_GROUP_IOSCHED=y; für die Aktivierung der Drosselungspolitik setzen Sie CONFIG_BLK_DEV_THROTTLING=y.
Weitere Informationen finden Sie unter Documentation/admin-guide/cgroup-v1/blkio-controller.rst.

1.31.4 CPU controller →

CONFIG_CGROUP_SCHED [=y] **[Y]**
Diese Funktion ermöglicht es dem CPU-Scheduler, Task-Gruppen zu erkennen und die Zuweisung von CPU-Bandbreite an solche Task-Gruppen zu steuern. Er verwendet cgroups, um Tasks zu gruppieren.

1.31.4.1 Group scheduling for SCHED_OTHER

CONFIG_FAIR_GROUP_SCHED [=y] **[Y]**
(keine Hilfe verfügbar)

1.31.4.1.1 CPU bandwidth provisioning for FAIR_GROUP_SCHED

CONFIG_CFS_BANDWIDTH [=y] **[Y]**
Mit dieser Option können Benutzer CPU-Bandbreitenraten (Limits) für Aufgaben festlegen, die innerhalb des Fair Group Schedulers laufen. Gruppen, für die kein Limit festgelegt wurde, gelten als uneingeschränkt und werden ohne Einschränkung ausgeführt.
Weitere Informationen finden Sie unter Documentation/scheduler/sched-bwc.rst.

1.31.4.2 Group scheduling for SCHED_RR/FIFO

CONFIG_RT_GROUP_SCHED [=n] **[N]**
Mit dieser Funktion können Sie den Task-Gruppen explizit echte CPU-Bandbreite zuweisen. Wenn sie aktiviert ist, wird es auch unmöglich, Echtzeitaufgaben für Nicht-Root-Benutzer zu planen, bis Sie ihnen Echtzeitbandbreite zuweisen.
Weitere Informationen finden Sie unter Documentation/scheduler/sched-rt-group.rst.

1.31.5 Utilization clamping per group of tasks

CONFIG_UCLAMP_TASK_GROUP [=y] **[Y]**
Mit dieser Funktion kann der Scheduler die geklemmte Auslastung jeder CPU auf der Grundlage der RUNNABLE-Tasks, die derzeit auf dieser CPU geplant sind, verfolgen. Wenn diese Option aktiviert ist, kann der Benutzer eine minimale und maximale CPU-Bandbreite angeben, die für jede einzelne Aufgabe in einer Gruppe zulässig ist. Mit der maximalen Bandbreite kann die maximale Frequenz, die ein Task verwenden kann, festgelegt werden, während mit der minimalen Bandbreite eine minimale Frequenz festgelegt werden kann, die ein Task immer verwenden wird. Bei aktivierter aufgabengruppenbasierter Auslastungsbegrenzung wird ein eventuell angegebener aufgabenspezifischer Begrenzungswert durch den von cgroup angegebenen Begrenzungswert eingeschränkt. Sowohl die minimale als auch die maximale Task-Klemmung kann nicht größer sein als die entsprechende auf Task-Gruppen-Ebene definierte Klemmung.
Im Zweifelsfall sagen Sie N.

1.31.6 PIDs controller

CONFIG_CGROUP_PIDS [=y] **[Y]**
Erzwingt die Begrenzung der Prozessanzahl im Bereich einer cgroup. Jeder Versuch, mehr Prozesse zu forken, als in der cgroup erlaubt sind, schlägt fehl. PIDs sind grundsätzlich eine globale Ressource, da es ziemlich trivial ist, eine PID-Erschöpfung zu erreichen, bevor man auch nur eine konservative kmemcg-Grenze erreicht. Infolgedessen ist es möglich, ein System zum Stillstand zu bringen, ohne durch andere cgroup-Richtlinien eingeschränkt zu werden. Der PID-Regler ist dafür ausgelegt, dies zu verhindern. Es sollte beachtet werden, dass organisatorische Operationen (wie z.B. das Anhängen an eine cgroup-Hierarchie) *nicht* durch den PIDs-Controller blockiert werden, da das PIDs-Limit nur die Fähigkeit eines Prozesses zum Forking, nicht aber zum Anhängen an eine cgroup beeinflusst.

1.31.7 RDMA controller

CONFIG_CGROUP_RDMA [=y] **[Y]**
Ermöglicht die Durchsetzung der vom IB-Stack definierten RDMA-Ressourcen. Es ist relativ einfach für Verbraucher, RDMA-Ressourcen zu erschöpfen, was dazu führen kann, dass Ressourcen für andere Verbraucher nicht mehr verfügbar sind. Der RDMA-Controller ist dafür ausgelegt, dies zu verhindern. Das Anhängen von Prozessen mit aktiven RDMA-Ressourcen an die cgroup-Hierarchie ist erlaubt, auch wenn die Grenze der Hierarchie überschritten werden kann.

1.31.8 Freezer controller

CONFIG_CGROUP_FREEZER [=y] **[Y]**
Ermöglicht das Einfrieren und Aufheben des Einfrierens aller Aufgaben in einer C-Group. Diese Option betrifft die ORIGINAL cgroup-Schnittstelle. Der cgroup2-Speicher-Controller enthält standardmäßig wichtige In-Kernel-Speicherverbraucher.
Wenn Sie cgroup2 verwenden, sagen Sie N.

1.31.9 HugeTLB controller

CONFIG_CGROUP_HUGETLB [=y] **[Y]**
Bietet eine cgroup-Steuerung für HugeTLB-Seiten. Wenn Sie dies aktivieren, können Sie die HugeTLB-Nutzung pro cgroup begrenzen. Die Begrenzung wird während eines Seitenfehlers durchgesetzt. Da HugeTLB keine Seitenrückforderung unterstützt, bedeutet die Durchsetzung des Limits zum Zeitpunkt des Seitenfehlers, dass die Anwendung ein SIGBUS-Signal erhält, wenn sie versucht, über das Limit hinaus auf HugeTLB-Seiten zuzugreifen. Dies setzt voraus, dass die Anwendung im Voraus weiß, wie viele HugeTLB-Seiten sie für ihre Nutzung benötigt. Die Kontrollgruppe wird im dritten Page-lru-Zeiger verfolgt. Dies bedeutet, dass wir die Steuergruppe nicht mit einer riesigen Seite von weniger als 3 Seiten verwenden können.

1.31.10 Cpuset controller

CONFIG_CPUSETS [=y] **[Y]**
Mit dieser Option können Sie CPUSETs erstellen und verwalten, die es ermöglichen, ein System dynamisch in Gruppen von CPUs und Speicherknoten zu partitionieren und Aufgaben zuzuweisen, die nur innerhalb dieser Gruppen ausgeführt werden. Dies ist vor allem auf großen SMP- oder NUMA-Systemen nützlich.
Sagen Sie N, wenn Sie unsicher sind.

1.31.10.1 Include legacy /proc/<pid>/cpuset file

CONFIG_PROC_PID_CPUSET [=y] **[Y]**
This option will let you create and manage CPUSETs which allow dynamically partitioning a system into sets of CPUs and Memory Nodes and assigning tasks to run only within those sets. This is primarily useful on large SMP or NUMA systems.
Say N if unsure.

1.31.11 Device controller

CONFIG_CGROUP_DEVICE [=y] **[Y]**
Bietet einen cgroup-Controller an, der Whitelists für Geräte implementiert, die ein Prozess in der cgroup mknod oder öffnen kann.

1.31.12 Simple CPU accounting controller

CONFIG_CGROUP_CPUACCT [=y] **[Y]**
(Einfacher CPU-Accounting-Controller)
Bietet einen einfachen Controller für die Überwachung des gesamten CPU-Verbrauchs der Tasks in einer cgroup an.

1.31.13 Perf controller

CONFIG_CGROUP_PERF [=y] **[Y]**
Diese Option erweitert den Modus perf per-cpu, um die Überwachung auf Threads zu beschränken, die zu der angegebenen cgroup gehören und auf der angegebenen CPU laufen. Sie kann auch verwendet werden, um die cgroup ID in Stichproben zu haben, so dass sie Leistungsereignisse zwischen cgroups überwachen kann.
Sagen Sie N, wenn Sie unsicher sind.

1.31.14 Support for eBPF programs attached to cgroups

CONFIG_CGROUP_BPF [=y] **[Y]**
Erlaubt das Anhängen von eBPF-Programmen an eine cgroup mit dem bpf(2)-Syscall-Befehl texttttBPF_PROG_ATTACH.
In welchem Kontext auf diese Programme zugegriffen wird, hängt von der Art des Attachments ab. Zum Beispiel werden Programme, die mit BPF_CGROUP_INET_INGRESS angehängt werden, auf dem Ingress-Pfad von inet-Sockets ausgeführt.

1.31.15 Misc resource controller

CONFIG_CGROUP_MISC [=y] **[Y]**
Bietet einen Controller für verschiedene Ressourcen auf einem Host. Verschiedene skalare Ressourcen sind die Ressourcen auf dem Host-System, die nicht wie die anderen cgroups abstrahiert werden können. Dieser Controller verfolgt und begrenzt die verschiedenen Ressourcen, die von einem Prozess verwendet werden, der an eine cgroup-Hierarchie angeschlossen ist.
Weitere Informationen finden Sie im Abschnitt misc cgroup in /Documentation/admin-guide/cgroup-v2.rst.

1.31.16 Debug controller

CONFIG_CGROUP_DEBUG [=n] **[N]**
Diese Option aktiviert einen einfachen Controller, der Debugging-Informationen über das cgroups-Framework exportiert. Dieser Controller ist nur für das Debugging von Kontroll-C-Gruppen gedacht. Seine Schnittstellen sind nicht stabil.
Sagen Sie N.

1.32 Namespaces support →

CONFIG_NAMESPACES [=y] **[Y]**
(Unterstützung von Namensräumen, namespaces)
Bietet die Möglichkeit, Aufgaben mit verschiedenen Objekten unter Verwendung derselben Kennung arbeiten zu lassen. Zum Beispiel kann sich dieselbe IPC-ID auf verschiedene Objekte beziehen oder dieselbe Benutzer-ID oder pid kann sich auf verschiedene Aufgaben beziehen, wenn sie in verschiedenen Namensräumen verwendet werden.

1.32.1 UTS namespace

CONFIG_UTS_NS [=y] **[Y]**
In diesem Namensraum sehen Aufgaben verschiedene Informationen, die mit dem Systemaufruf uname() bereitgestellt werden

1.32.2 TIME namespace

CONFIG_TIME_NS [=y] **[Y]**
In diesem Namespace können boottime und monotone Uhren eingestellt werden. Die Zeit läuft dann mit der gleichen Geschwindigkeit weiter.

1.32.3 IPC namespace

CONFIG_IPC_NS [=y] **[Y]**
In diesem Namensraum arbeiten Aufgaben mit IPC-IDs (Interprozess-IDs), die jeweils verschiedenen IPC-Objekten in verschiedenen Namensräumen entsprechen.

1.32.4 User namespace

CONFIG_USER_NS [=y] **[Y]**
Dies ermöglicht es Containern, d.h. V-Servern, Benutzernamensräume zu verwenden, um verschiedene Benutzerinformationen für verschiedene Server bereitzustellen. Wenn Benutzernamensräume im Kernel aktiviert sind, wird empfohlen, dass die Option `MEMCG` ebenfalls aktiviert wird und dass der Benutzerbereich die Speicherkontrollgruppen verwendet, um die Speichermenge zu begrenzen, die nicht privilegierte Benutzer verwenden können.

1.32.4.1 Allow unprivileged users to create namespaces

CONFIG_USERS_NS_UNPRIVILEGED [=y] **[Y]**
Wenn diese Funktion deaktiviert ist, können unprivilegierte Benutzer keine neuen Namensräume erstellen. Die Möglichkeit, dass Benutzer ihre eigenen Namespaces erstellen können, war Teil mehrerer kürzlich erfolgter lokaler Privilegienerweiterungen. Wenn Sie also Benutzernamespaces benötigen, aber paranoid bzw. sicherheitsbewusst sind, sollten Sie diese Funktion deaktivieren. Diese Einstellung kann zur Laufzeit mit dem `kernel.unprivileged_userns_clone sysctl` außer Kraft gesetzt werden.
Wenn Sie unsicher sind, sagen Sie Y.

1.32.5 PID namespace

CONFIG_PID_NS [=y] **[Y]**
Unterstützung von Prozess-ID-Namensräumen. Dies ermöglicht es, mehrere Prozesse mit der gleichen pid zu haben, solange sie sich in verschiedenen pid-Namensräumen befinden. Dies ist ein Baustein von Containern.

1.32.6 Network namespace

CONFIG_NET_NS [=y] **[Y]**
Ermöglicht es dem Benutzer, scheinbar mehrere Instanzen des Netzwerkstapels zu erstellen.

1.33 Checkpoint/restore support

CONFIG_CHECKPOINT_RESTORE [=y] **[Y]**
Ermöglicht zusätzliche Kernel-Funktionen in einer Art Checkpoint/Restore. Insbesondere fügt es zusätzliche prctl-Codes zum Einrichten von Prozesstext, Daten- und Heap-Segmentgrößen sowie einige zusätzliche /proc-Dateisystemeinträge hinzu.
Wenn Sie unsicher sind, geben Sie hier N an.

1.34 Automatic process group scheduling

CONFIG_SCHED_AUTOGROUP [=y] **[Y]**
Mit dieser Option wird der Scheduler für gängige Desktop-Workloads optimiert, indem automatisch Aufgabengruppen erstellt und aufgefüllt werden. Diese Trennung von Arbeitslasten isoliert aggressive CPU-Brenner (wie Build-Jobs) von Desktop-Anwendungen. Die automatische Erstellung von Aufgabengruppen basiert derzeit auf der Aufgabensitzung.

1.35 Kernel→user space relay support (formerly relayfs)

CONFIG_RELAY [=y] **[Y]**
Diese Option aktiviert die Unterstützung für die Relaisschnittstelle in bestimmten Dateisystemen (wie debugfs). Sie wurde entwickelt, um einen effizienten Mechanismus für Werkzeuge und Einrichtungen zur Weiterleitung großer Datenmengen aus dem Kernelbereich in den Benutzerbereich bereitzustellen.
Wenn Sie unsicher sind, sagen Sie N.

1.36 Initial RAM filesystem and RAM disk (initramfs/initrd) support

CONFIG_BLK_DEV_INITRD [=y] **[Y]**
Das anfängliche RAM-Dateisystem ist ein ramfs, das vom Bootloader (loadlin oder lilo) geladen und vor dem normalen Bootvorgang als root eingehängt wird. Es wird typischerweise verwendet, um Module zu laden, die zum Einhängen des „echten“ Root-Dateisystems benötigt werden, usw.
Siehe <file:Documentation/admin-guide/initrd.rst> für Details. Wenn die RAM-Disk-Unterstützung (BLK_DEV_RAM) ebenfalls enthalten ist, aktiviert dies auch die anfängliche RAM-Disk-Unterstützung (initrd) und fügt 15 KByte (auf einigen anderen Architekturen mehr) zur Kernelgröße hinzu.
Wenn Sie unsicher sind, sagen Sie Y.

1.36.1 Initramfs source file(s)

CONFIG_INITRAMFS_SOURCE [=] [ ]
Dies kann entweder ein einzelnes cpio-Archiv mit der Endung .cpio oder eine durch Leerzeichen getrennte Liste von Verzeichnissen und Dateien zur Erstellung des initramfs-Abbilds sein. Ein cpio-Archiv sollte ein Dateisystemarchiv enthalten, das als initramfs-Abbild verwendet werden soll. Verzeichnisse sollten ein Dateisystem-Layout enthalten, das in das initramfs-Abbild aufgenommen werden soll. Die Dateien sollten Einträge in dem Format enthalten, das vom Programm `usr/gen_init_cpio` im Kernelbaum beschrieben wird. Wenn mehrere Verzeichnisse und Dateien angegeben werden, wird das initramfs-Abbild die Summe aller dieser Verzeichnisse und Dateien sein.
Siehe <file:Documentation/driver-api/early-userspace/early_userspace_support.rst> für weitere Details.
Wenn Sie sich nicht sicher sind, lassen Sie das Feld leer.
Symbol: INITRAMFS_SOURCE [=]
Type : string (Zeichenkette)

1.36.2 Support initial ramdisk/ramfs compressed using gzip

CONFIG_RD_GZIP [=y] **[Y]**
Unterstützung des Ladens eines gzip-kodierten Anfangs-Ramdisk- oder Cpio-Puffers.
Wenn Sie unsicher sind, sagen Sie Y.

1.36.3 Support initial ramdisk/ramfs compressed using bzip2

CONFIG_RD_BZIP2 [=y] **[Y]**
Unterstützung des Ladens eines bzip2-kodierten Anfangs-Ramdisk- oder Cpio-Puffers.
Wenn Sie unsicher sind, sagen Sie Y.

1.36.4 Support initial ramdisk/ramfs compressed using LZMA

CONFIG_RD_LZMA [=y] **[Y]**
Unterstützung des Ladens eines LZMA-kodierten Anfangs-Ramdisk- oder Cpio-Puffers.
Wenn Sie unsicher sind, sagen Sie Y.

1.36.5 Support initial ramdisk/ramfs compressed using XZ

CONFIG_RD_XZ [=y] **[Y]**
Unterstützung des Ladens eines XZ-kodierten Anfangs-Ramdisk- oder Cpio-Puffers.
Wenn Sie unsicher sind, sagen Sie Y.

1.36.6 Support initial ramdisk/ramfs compressed using LZO

CONFIG_RD_LZO [=y] **[Y]**
Unterstützung des Ladens eines LZO-kodierten Anfangs-Ramdisk- oder Cpio-Puffers.
Wenn Sie unsicher sind, sagen Sie Y.

1.36.7 Support initial ramdisk/ramfs compressed using LZ4

CONFIG_RD_LZ4 [=y] **[Y]**
Unterstützung des Ladens eines LZ4-kodierten Anfangs-Ramdisk- oder Cpio-Puffers.
Wenn Sie unsicher sind, sagen Sie Y.

1.36.8 Support initial ramdisk/ramfs compressed using ZSTD

CONFIG_RD_ZSTD [=y] **[Y]**
Unterstützung des Ladens eines ZSTD-kodierten Anfangs-Ramdisk- oder Cpio-Puffers.
Wenn Sie unsicher sind, sagen Sie Y.

1.37 Boot config support

CONFIG_BOOT_CONFIG [=y] **[Y]**
Extra boot config ermöglicht es dem Systemadministrator, eine Konfigurationsdatei als zusätzliche Erweiterung der Kernel-Cmdline beim Booten zu übergeben. Die Bootkonfigurationsdatei muss am Ende von initramfs mit Prüfsumme, Größe und magischem Wort angehängt werden.
Siehe <file:Documentation/admin-guide/bootconfig.rst> für Details.
Wenn Sie unsicher sind, sagen Sie Y.

1.37.1 Force unconditional bootconfig processing

CONFIG_BOOT_CONFIG_FORCE [=n] **[N]**
Wenn diese Kconfig-Option gesetzt ist, wird die BOOT_CONFIG-Verarbeitung auch dann durchgeführt, wenn der Kernel-Boot-Parameter "bootconfig"weggelassen wird. Tatsächlich gibt es mit dieser Kconfig-Option keine Möglichkeit, den Kernel dazu zu bringen, die von BOOT_CONFIG gelieferten Kernel-Boot-Parameter zu ignorieren.
Wenn Sie unsicher sind, sagen Sie N.

1.37.2 Embed bootconfig file in the kernel

CONFIG_BOOT_CONFIG_EMBED [=n] **[N]**
Eine mit BOOT_CONFIG_EMBED_FILE angegebene bootconfig-Datei in den Kernel einbetten. Normalerweise wird die bootconfig-Datei mit dem initrd-Image geladen. Wenn das System jedoch initrd nicht unterstützt, hilft Ihnen diese Option, indem sie eine bootconfig-Datei beim Erstellen des Kernels einbettet.
Wenn Sie unsicher sind, sagen Sie N.

1.38 Preserve cpio archive mtimes in initramfs

CONFIG_INITRAMFS_PRESERVE_MTIME [=y] **[Y]**
Jeder Eintrag in einem initramfs cpio-Archiv enthält einen mtime-Wert. Wenn diese Option aktiviert ist, übernehmen die extrahierten cpio-Einträge diese mtime, wobei die mtime-Einstellung des Verzeichnisses aufgeschoben wird, bis nach der Erstellung aller untergeordneten Einträge.
Wenn Sie unsicher sind, sagen Sie Y.

1.39 Compiler optimization level →

Optimierungsgrad des Compilers, Auswahl aus den folgenden zwei Punkten:

1.39.1 Optimize for performance (-O2)

CONFIG_CC_OPTIMIZE_FOR_Performance [=y] **[Y]**
Dies ist die Standardoptimierungsstufe für den Kernel, die mit dem Compiler-Flag `-O2` erstellt wird, um die beste Leistung und die hilfreichsten Warnungen bei der Kompilierung zu erhalten.

1.39.2 Optimize for size (-Os)

CONFIG_CC_OPTIMIZE_FOR_SIZE [=n] **[N]**
Wenn Sie diese Option wählen, wird `-Os` an Ihren Compiler übergeben, was zu einem kleineren Kernel führt.

1.40 Configure standard kernel features (expert users)

CONFIG_EXPERT [=n] [ ]
Mit dieser Option können bestimmte Basis-Kerneloptionen und -einstellungen deaktiviert oder optimiert werden. Dies ist für spezielle Umgebungen gedacht, die einen „Nicht-Standard“-Kernel tolerieren können. Verwenden Sie diese Option nur, wenn Sie wirklich wissen, was Sie tun.

1.40.1 Load all symbols for debugging/ksymoops

CONFIG_KALLSYMS [=y] **[Y]**
(sichtbar wenn EXPERT [=n])
Geben Sie hier Y ein, damit der Kernel symbolische Absturzinformationen und symbolische Stack-Backtraces ausgibt. Dies erhöht die Größe des Kernels etwas, da alle Symbole in das Kernel-Image geladen werden müssen.

1.40.1.1 Test the basic functions and performance of kallsyms

CONFIG_KALLSYMS_SELFTEST [=n] **[N]**
Testen Sie die Grundfunktionen und die Leistung einiger Schnittstellen, wie z. B. `kallsyms_lookup_name`. Außerdem wird die Kompressionsrate des kallsyms-Kompressionsalgorithmus für den aktuellen Symbolsatz berechnet. Starten Sie den Selbsttest automatisch nach dem Systemstart.
Es wird empfohlen, `dmesg | grep kallsyms_selftest` auszuführen, um die Testergebnisse zu sammeln. In der letzten Zeile wird `finish` angezeigt, was bedeutet, dass der Test abgeschlossen ist.

1.40.1.2 Include all symbols in kallsyms

CONFIG_KALLSYMS_ALL [=y] **[Y]**
Normalerweise enthält kallsyms nur die Symbole von Funktionen für schönere OOPS-Meldungen und Backtraces (d. h. Symbole aus den Abschnitten text und inittext). Dies ist für die meisten Fälle ausreichend. Nur wenn Sie Kernel-Live-Patching oder andere weniger häufige Anwendungsfälle (z. B. wenn ein Debugger verwendet wird) aktivieren wollen, sind alle Symbole erforderlich (d. h. die Namen von Variablen aus den Data-Abschnitten usw.).
Diese Option stellt sicher, dass alle Symbole in das Kernel-Image geladen werden (d.h. Symbole aus allen Sektionen), was die Kernelgröße erhöht (je nach Kernelkonfiguration kann sie 300KiB oder etwas Ähnliches betragen).
Sagen Sie N, es sei denn, Sie brauchen wirklich alle Symbole, oder Kernel-Live-Patching.

1.41 Kernel Performance Events And Counters →

Kernel-Leistungsereignisse und -Zähler

1.41.1 Kernel performance events and counters

CONFIG_PERF_EVENTS [=y] **[Y]**
Aktivieren Sie die Kernel-Unterstützung für verschiedene von Software und Hardware bereitgestellte Leistungsereignisse.

Software-Ereignisse werden entweder integriert oder über die Verwendung von generischen Tracepoints unterstützt.

Die meisten modernen CPUs unterstützen Leistungsereignisse über Leistungszählerregister. Diese Register zählen die Anzahl bestimmter Arten von hw-Ereignissen: z. B. ausgeführte Anweisungen, erlittene Cachemisses oder falsch vorhergesagte Verzweigungen – ohne den Kernel oder Anwendungen zu verlangsamen. Diese Register können auch Unterbrechungen auslösen, wenn eine bestimmte Anzahl von Ereignissen überschritten wird – und können so dazu verwendet werden, ein Profil des Codes zu erstellen, der auf dieser CPU läuft.

Das Linux-Performance-Event-Subsystem bietet eine Abstraktion dieser Software- und Hardware-Event-Fähigkeiten, die über einen Systemaufruf zugänglich sind und von dem Dienstprogramm `perf` in `tools/perf/` verwendet werden. Es stellt Zähler pro Task und pro CPU zur Verfügung und bietet darüber hinaus Ereignisfunktionen.
Sagen Sie Y, wenn Sie unsicher sind.

1.41.1.1 Debug: use vmalloc to back perf mmap() buffers

CONFIG_DEBUG_PERF_USE_VMALLOC [=n] **[N]**
Verwendung von vmalloc-Speicher zur Sicherung von mmap()-Puffern. Hauptsächlich nützlich zum Debuggen des vmalloc-Codes auf Plattformen, die dies nicht erfordern. Sagen Sie N, wenn Sie unsicher sind.

1.42 Profiling support

CONFIG_PROFILING [=y] **[Y]**
Sagen Sie hier Y, um die erweiterten Unterstützungsmechanismen für das Profiling zu aktivieren, die von Profilern verwendet werden.

1.43 Kexec and crash features →

Kexec und Absturzmerkmale

1.43.1 Enable kexec system call

CONFIG_KEXEC [=y] **[Y]**
`kexec` ist ein Systemaufruf, der die Fähigkeit implementiert, den aktuellen Kernel herunterzufahren und einen anderen Kernel zu starten. Es ist wie ein Neustart, aber er ist unabhängig von der System-Firmware. Und wie ein Neustart können Sie damit jeden Kernel starten, nicht nur Linux. Der Name kommt von der Anlehnung mit dem Systemaufruf `exec`. Es ist ein fortlaufender Prozess, um sicher zu sein, dass die Hardware eines Rechners ordnungsgemäß heruntergefahren wird, seien Sie also nicht überrascht, wenn dieser Code bei Ihnen zunächst nicht funktioniert. Zum Zeitpunkt des Verfassens dieses Artikels ist die genaue Hardwareschnittstelle noch stark im Wandel, so dass keine gute Empfehlung ausgesprochen werden kann.

1.43.2 Enable kexec file based system call

CONFIG_KEXEC_FILE [=y] **[Y]**
(Aktivieren des dateibasierten Systemaufrufs kexec)
Dies ist eine neue Version des Systemaufrufs `kexec`. Dieser Systemaufruf ist dateibasiert und nimmt Dateideskriptoren als Systemaufrufsargument für Kernel und initramfs anstelle einer Liste von Segmenten, wie sie vom kexec-Systemaufruf akzeptiert wird.

1.43.2.1 Verify kernel signature during kexec_file_load() syscall

CONFIG_KEXEC_SIG [=y] **[Y]**
Mit dieser Option wird der Syscall `kexec_file_load()` auf eine gültige Signatur des Kernel-Images geprüft. Das Image kann immer noch ohne gültige Signatur geladen werden, es sei denn, Sie aktivieren auch KEXEC_SIG_FORCE, aber wenn es eine Signatur gibt, die überprüft werden kann, dann muss sie auch gültig sein. Zusätzlich zu dieser Option müssen Sie die Signaturprüfung für den entsprechenden Kernel-Image-Typ, der geladen wird, aktivieren, damit dies funktioniert.

1.43.2.1.1 Require a valid signature in kexec_file_load() syscall

CONFIG_KEXEC_SIG [=n] **[N]**
Diese Option macht die Überprüfung der Kernelsignatur für den Syscall `kexec_file_load()` zwingend erforderlich.

1.43.2.1.2 Enable bzImage signature verification support

CONFIG_KEXEC_BZIMAGE_VERIFY_SIG [=n] **[N]**
Aktivierung der Unterstützung von bzImage für die Signaturprüfung.

1.43.3 kexec jump

CONFIG_KEXEC_JUMP [=y] **[Y]**
Sprung zwischen Original-Kernel und kexeced-Kernel und Aufruf von Code im physikalischen Adressmodus über KEXEC

1.43.4 kexec crash dumps

CONFIG_KEXEC_DUMP [=y] **[Y]**
Absturzdump (Speicherauszug) erzeugen, nachdem er von kexec gestartet wurde. Dies sollte normalerweise nur in speziellen Crash-Dump-Kerneln gesetzt werden, die im Hauptkernel mit kexec-tools in einen speziell reservierten Bereich geladen werden und dann später nach einem Absturz von kdump/kexec ausgeführt werden. Der Crash-Dump-Kernel muss mit PHYSICAL_START auf eine Speicheradresse kompiliert werden, die nicht vom Hauptkernel oder BIOS verwendet wird, oder er muss als relocatable image (CONFIG_RELOCATABLE=y) erstellt werden.
Für weitere Details siehe Documentation/admin-guide/kdump/kdump.rst

Für s390 aktiviert diese Option auch zfcpdump.
Siehe auch <file:Documentation/s390/zfcpdump.rst>

1.43.4.1 Update the crash elfcorehdr on system configuration changes

CONFIG_CRASH_HOTPLUG [=y] **[Y]**
Aktivierung der direkten Aktualisierung der Crash-Elfcorehdr (die die Liste der CPUs und Speicherbereiche enthält, die bei einem Absturz gelöscht werden sollen) als Reaktion auf Hot-Plug/Unplug oder Online/Offline von CPUs oder Speicher. Dies ist ein sehr viel fortschrittlicherer Ansatz als der Versuch dies im Userspace zu tun.
Wenn Sie unsicher sind, sagen Sie Y.

1.43.4.1.1 Specify the maximum number of memory regions for the elfcorehdr

CONFIG_CRASH_MAX_MEMORY_RANGES [=8192] **[8192]**
Für den Pfad des Systemaufrufs textttkexec_file_load() ist die maximale Anzahl der Speicherbereiche anzugeben, die der elfcorehdr-Puffer/das elfcorehdr-Segment aufnehmen kann. Diese Regionen werden über textttwalk_system_ram_res() ermittelt, z.B. die 'System RAM'-Einträge in /proc/iomem. Dieser Wert wird mit NR_CPUS_DEFAULT kombiniert und mit `sizeof(Elf64_Phdr)` multipliziert, um die endgültige elfcorehdr-Speicherpuffer-/Segmentgröße zu bestimmen. Der Wert 8192 beispielsweise deckt ein (dünn besiedeltes) 1TiB-System ab, das aus 128MiB-Memblöcken besteht, und führt zu einer elfcorehdr-Speicherpuffer-/Segmentgröße von unter 1MiB. Dies ist eine vernünftige Wahl, um sowohl Baremetal- als auch virtuelle Maschinenkonfigurationen zu unterstützen.
Für den Syscall-Pfad `kexec_load()` ist CRASH_MAX_MEMORY_RANGES Teil der Berechnung hinter dem Wert, der über das Attribut /sys/kernel/crash_elfcorehdr_size bereitgestellt wird.

2 64-bit kernel

CONFIG_64BIT [=y] **[Y]**
Sagen Sie Y für ja, zur Erstellung eines 64-Bit-Kernels - früher bekannt als x86_64
Sagen Sie N für nein, um einen 32-Bit-Kernel zu erstellen - früher bekannt als i386

3 Processor type and features →

Prozessortyp und Eigenschaften

3.1 Symmetric multi-processing support

CONFIG_SMP [=y] **[Y]**
Dies ermöglicht die Unterstützung von Systemen mit mehr als einer CPU. Wenn Sie ein System mit nur einer CPU haben, sagen Sie N. Wenn Sie ein System mit mehr als einer CPU haben, sagen Sie Y. Wenn Sie hier N angeben, läuft der Kernel auf Uni- und Multiprozessor-Maschinen, verwendet aber nur eine CPU einer Multiprozessor-Maschine. Wenn Sie hier Y angeben, läuft der Kernel auf vielen, aber nicht auf allen Uniprozessor-Maschinen.

Auf einer Uniprozessor-Maschine läuft der Kernel schneller, wenn Sie hier N angeben. Beachten Sie, dass der Kernel nicht auf 486er-Architekturen läuft, wenn Sie hier Y angeben und unter „Prozessorfamilie“ die Architektur „586“ oder „Pentium“ auswählen.

Ebenso funktionieren Multiprozessor-Kernel für die „PPro“-Architektur möglicherweise nicht auf allen Pentium-basierten Boards.

Benutzer von Multiprozessor-Maschinen, die hier „Ja“ angeben, sollten auch „Ja“ zu „Enhanced Real Time Clock Support“ (siehe unten) sagen. Der „Advanced Power Management“-Code wird deaktiviert, wenn Sie hier „Y“ angeben. Siehe auch <file:Documentation/arch/x86/i386/IO-APIC.rst>, <file:Documentation/admin-guide/lockup-watchdogs.rst> und das SMP-HOWTO, verfügbar unter: `http://www.tldp.org/docs.html#howto`.
Wenn Sie nicht wissen, was Sie hier tun sollen, sagen Sie N.

3.2 Support x2apic

CONFIG_X86_X2APIC [=y] **[Y]**
Dies ermöglicht die Unterstützung von x2apic auf CPUs, die über diese Funktion verfügen. Dies ermöglicht 32-Bit-Apic-IDs (so dass es sehr große Systeme unterstützen kann) und greift auf den lokalen apic über MSRs und nicht über mmio zu. Einige Intel-Systeme ab ca. 2022 sind in den x2APIC-Modus gesperrt und können nicht auf die alten APIC-Modi zurückgreifen, wenn SGX oder TDX im BIOS aktiviert sind. Ohne Aktivierung dieser Option booten sie mit stark eingeschränkter Funktionalität.
Wenn Sie nicht wissen, was Sie hier tun sollen, sagen Sie N.

3.3 Enable MPS table

CONFIG_X86_MPPARSE [=y] **[Y]**
Für alte smp-Systeme, die keine richtige acpi-Unterstützung haben. Neuere Systeme (insbesondere mit 64bit-CPUs) mit acpi-Unterstützung, werden von MADT und DSDT überschrieben.

3.4 x86 CPU resource control support

CONFIG_X86_CPU_RESCTRL [=y] **[Y]**
Aktivieren Sie die Unterstützung der x86-CPU-Ressourcensteuerung. Unterstützung für die Zuweisung und Überwachung der Nutzung von Systemressourcen durch die CPU. Intel nennt dies Intel Resource Director Technology (Intel(R) RDT). Weitere Informationen über RDT finden Sie im Intel x86 Architecture Software Developer Manual. AMD bezeichnet dies als AMD Platform Quality of Service (AMD QoS).
Weitere Informationen über AMD QoS finden Sie im Handbuch AMD64 Technology Platform Quality of Service Extensions.
Sagen Sie N, wenn Sie unsicher sind.

3.5 Support for extended (non-PC) x86 platforms

CONFIG_X86_EXTENDED_PLATFORM [=n] **[N]**
Wenn Sie diese Option deaktivieren, unterstützt der Kernel nur Standard-PC-Plattformen (was die große Mehrheit der Systeme da draußen abdeckt). Wenn Sie diese Option aktivieren, können Sie die Unterstützung für die folgenden (nicht-PC) 64-Bit-x86-Plattformen auswählen:

- Numascale NumaChip
- ScaleMP vSMP
- SGI Ultraviolet

Wenn Sie eines dieser Systeme haben, oder wenn Sie einen generischen Distributionskernel bauen wollen, geben Sie hier Y an – andernfalls sagen Sie N.

3.6 Intel Low Power Subsystem Support

CONFIG_X86_INTEL_LPSS [=y] **[Y]**
Wählen Sie diese Option, um Unterstützung für das Intel Low Power Subsystem zu erstellen, wie es auf dem Intel Lynxpoint PCH zu finden ist. Die Auswahl dieser Option ermöglicht Dinge wie Clock Tree (Common Clock Framework) und Pincontrol, die von den LPSS-Peripherietreibern benötigt werden.

3.7 AMD ACPI2Platform devices support

CONFIG_X86_AMD_PLATFORM_DEVICE [=y] **[Y]**
Wählen Sie diese Option, um AMD-spezifische ACPI-Geräte wie I2C, UART, GPIO, die auf AMD Carrizo und späteren Chipsätzen zu finden sind, als Plattformgeräte zu interpretieren. I2C und UART hängen von COMMON_CLK ab, um den Takt zu setzen. Der GPIO-Treiber ist im PINCTRL-Subsystem implementiert.

3.8 Intel SoC IOSF Sideband support for SoC platforms

CONFIG_IOSF_MBI [=y] **[Y]**
Diese Option aktiviert die Unterstützung des Seitenband-Registerzugriffs für Intel SoC-Plattformen. Auf diesen Plattformen wird das IOSF-Seitenband anstelle von MSRs für einige Registerzugriffe verwendet, vor allem, aber nicht ausschließlich, für thermische und Stromversorgungs-Register. Treiber können die Verfügbarkeit dieser Geräte abfragen, um festzustellen, ob sie das Seitenband benötigen, um auf diesen Plattformen zu funktionieren. Das Seitenband ist auf den folgenden SoC-Produkten verfügbar.

- BayTrail
- Braswell
- Quark

Sie sollten Y sagen, wenn Sie einen Kernel auf einem dieser SoCs ausführen.

3.8.1 Enable IOSF sideband access through debugfs

CONFIG_IOSF_MBI_DEBUG [=n] **[N]**
Wählen Sie diese Option, um die IOSF-Seitenband-Zugriffsregister (MCR, MDR, MCRX) über debugfs freizugeben, um Registerinformationen von verschiedenen Einheiten auf dem SoC zu schreiben und zu lesen. Dies ist sehr nützlich, um Informationen über den Gerätezustand für Debugging und Analyse zu erhalten. Da es sich um einen allgemeinen Zugriffsmechanismus handelt, müssen die Benutzer dieser Option das Gerät, auf das sie zugreifen wollen, genau kennen.
Wenn Sie die Option nicht benötigen oder im Zweifel sind, sagen Sie N.

3.9 Single-depth WCHAN output

CONFIG_SHED_OMIT_FRAME_POINTER [=y] **[Y]**
Berechne einfachere /proc/<PID>/wchan-Werte. Wenn diese Option deaktiviert ist, werden die wchan-Werte zur aufrufenden Funktion zurückgeführt. Dies liefert genauere wchan-Werte, allerdings auf Kosten eines etwas größeren Planungsaufwands (scheduling overhead).
Im Zweifelsfall sagen Sie "Y".

3.10 Linux guest support →

CONFIG_HYPERVISOR_GUEST [=y] **[Y]**
Geben Sie hier Y ein, um Optionen für die Ausführung von Linux unter verschiedenen Hypervisoren zu aktivieren. Diese Option aktiviert die grundlegende Hypervisor-Erkennung und die Einrichtung der Plattform. Wenn Sie N sagen, werden alle Optionen in diesem Untermenü übersprungen und deaktiviert, und die Linux-Gastunterstützung wird nicht eingebaut.

3.10.1 Enable paravirtualization code

CONFIG_PARAVIRT [=y] **[Y]**
Der Kernel wird so verändert, dass er sich selbst modifizieren kann, wenn er unter einem Hypervisor ausgeführt wird, was die Leistung gegenüber einer vollständigen Virtualisierung erheblich verbessern kann. Wenn der Kernel jedoch ohne Hypervisor ausgeführt wird, ist er theoretisch langsamer und etwas größer.

3.10.2 paravirt-ops debugging

CONFIG_PARAVIRT_DEBUG [=n] **[N]**
Ermöglicht das Debuggen von paravirt_ops Interna. Insbesondere BUG, wenn eine paravirt_op fehlt, wenn sie aufgerufen wird.

3.10.3 Paravirtualization layer for spinlocks

CONFIG_PARAVIRT_SPINLOCKS [=y] **[Y]**
Paravirtualisierte Spinlocks ermöglichen es einem pvops-Backend, die Spinlock-Implementierung durch etwas Virtualisierungsfreundliches zu ersetzen (z. B. Blockieren der virtuellen CPU anstelle von Spinning). Dies hat nur minimale Auswirkungen auf native Kernel und bringt einen deutlichen Leistungsvorteil für paravirtualisierte KVM/Xen-Kernel.
Wenn Sie unsicher sind, wie Sie diese Frage beantworten sollen, antworten Sie mit Y.

3.10.4 Xen guest support

CONFIG_XEN [=y] **[Y]**
Dies ist der Linux-Xen-Port. Wenn Sie dies aktivieren, kann der Kernel in einer paravirtualisierten Umgebung unter dem Xen-Hypervisor booten.

3.10.4.1 Xen PV guest support

CONFIG_XEN_PV [=y] **[Y]**
Der Betrieb als Xen PV-Gast wird unterstützt.

3.10.4.1.1 Limit Xen pv-domain memory to 512GB

CONFIG_XEN_512GB [=y] **[Y]**
Begrenzen der paravirtualisierten Benutzerdomänen auf 512 GB RAM. Die Xen-Tools und die Tools zur Analyse von Crash-Dumps unterstützen möglicherweise keine pv-Domänen mit mehr als 512 GB RAM. Diese Option steuert die Standardeinstellung des Kernels, um nur bis zu 512 GB oder mehr zu verwenden. Es ist jederzeit möglich, die Standardeinstellung durch Angabe des Boot-Parameters `xen_512gb_limit` zu ändern.

3.10.4.2 Xen PVHVM guest support

CONFIG_XEN_PVHVM_GUEST [=y] **[Y]**
Der Betrieb als Xen PVHVM-Gast wird unterstützt.

3.10.4.3 Enable Xen debug and tuning parameters in debugfs

CONFIG_XEN_DEBUG_FS [=n] **[N]**
Der Betrieb als Xen PV-Gast wird unterstützt.

3.10.4.4 Xen PVH guest support

CONFIG_XEN_PVH [=y] **[Y]**
Der Betrieb als Xen PVH-Gast wird unterstützt.

3.10.5 Xen Dom0 support

CONFIG_XEN_DOM0 [=y] **[Y]**
Der Betrieb als Xen Dom0-Gast wird unterstützt.

3.10.6 Always use safe MSR accesses in PV guests

CONFIG_XEN_PV_MSR_SAFE [=y] **[Y]**
Verwenden Sie sichere (nicht fehlerhafte) MSR-Zugriffsfunktionen, auch wenn der MSR-Zugriff ohnehin nicht fehlerhaft sein sollte. Der Standardwert kann mit dem Boot-Parameter `xen_msr_safe` geändert werden.

3.10.7 KVM Guest support (including kvmclock)

CONFIG_KVM_GUEST [=y] **[Y]**
Diese Option ermöglicht verschiedene Optimierungen für die Ausführung unter dem KVM-Hypervisor. Sie beinhaltet eine paravirtualisierte Uhr, so dass der Host dem Gast eine Zeitinfrastruktur wie die Tageszeit und die Systemzeit zur Verfügung stellt, anstatt sich auf eine PIT-Emulation (oder wahrscheinlich eine andere) durch das zugrunde liegende Gerätemodell zu verlassen.

3.10.8 Disable host haltpoll when loading haltpoll driver

CONFIG_ARCH_CPUIDLE_HALTPOLL [=y] **[Y]**
(Haltpoll des Hosts beim Laden des Haltpoll-Treibers deaktivieren)
Wenn Sie unter KVM virtualisieren, deaktiviert den haltpoll des Hosts.

3.10.9 Support for running PVH guests

CONFIG_PVH [=y] **[Y]**
Diese Option aktiviert den PVH-Einstiegspunkt für virtuelle Gastmaschinen, wie in der x86/HVM Direct Boot ABI angegeben.

3.10.10 Paravirtual steal time accounting

CONFIG_PARAVIRT_TIME_ACCOUNTING [=y] **[Y]**
Wählen Sie diese Option aus, um die Berechnung der Zeit für das Stehlen von Aufgaben mit feiner Granularität zu aktivieren. Die Zeit, die für die Ausführung anderer Aufgaben parallel zur aktuellen vCPU aufgewendet wird, ist von der vCPU-Leistung abgezogen. Um dies zu berücksichtigen, kann es zu geringen Leistungseinbußen kommen.
Im Zweifelsfall geben Sie hier N an.

3.10.11 Jailhouse non-root cell support

CONFIG_JAILHOUSE_GUEST [=y] **[Y]**
Diese Option ermöglicht es, Linux als Gast in einer Jailhouse-Nicht-Root-Zelle auszuführen. Sie können diese Option deaktiviert lassen, wenn Sie Jailhouse nur starten und Linux anschließend in der Root-Zelle ausführen möchten.

3.10.12 ACRN Guest support

CONFIG_ACRN_GUEST [=y] **[Y]**
Diese Option ermöglicht die Ausführung von Linux als Gast im ACRN-Hypervisor. ACRN ist ein flexibler, leichtgewichtiger Referenz-Open-Source-Hypervisor, der mit Blick auf Echtzeit und Sicherheitskritik entwickelt wurde. Er wurde für eingebettete IOT mit kleinem Platzbedarf und Echtzeitfunktionen entwickelt. Weitere Einzelheiten finden Sie unter `https://projectacrn.org/`.

3.10.13 Intel TDX (Trust Domain Extensions) - Guest Support

CONFIG_INTEL_TDX_GUEST [=y] **[Y]**
Unterstützung der Ausführung als Gast unter Intel TDX.
Ohne diese Unterstützung kann der Gastkernel nicht booten oder unter TDX laufen. TDX umfasst Speicherverschlüsselungs- und Integritätsfunktionen, die die Vertraulichkeit und Integrität des Gastspeicherinhalts und des CPU-Status schützen. TDX-Gäste sind vor einigen Angriffen durch den VMM geschützt.

3.11 Processor family (Generic-x86-64) →

Dies ist der Prozessortyp Ihrer CPU. Diese Information wird für Optimierungszwecke verwendet. Um einen Kernel zu kompilieren, der auf allen unterstützten x86-CPU-Typen laufen kann (wenn auch nicht optimal schnell), können Sie hier „486“ angeben. Beachten Sie, dass der 386er nicht mehr unterstützt wird, dies schließt AMD/Cyrix/Intel 386DX/DXL/SL/SLC/SX, Cyrix/TI 486DLC/DLC2, UMC 486SX-S und den NexGen Nx586 ein. Der Kernel läuft nicht notwendigerweise auf älteren Architekturen als der von Ihnen gewählten, z.B. läuft ein Pentium-optimierter Kernel auf einem PPro, aber nicht unbedingt auf einem i486.

Hier sind die empfohlenen Einstellungen für höchste Geschwindigkeit:

- `486` für den AMD/Cyrix/IBM/Intel 486DX/DX2/DX4 oder SL/SLC/SLC2/SLC3/SX/SX2 und UMC U5D oder U5S.
- `586` für generische Pentium-CPUs, denen das TSC-Register (Zeitstempelzähler) fehlt.
- `Pentium-Classic` für den Intel Pentium.
- `Pentium-MMX` für den Intel Pentium MMX.
- `Pentium-Pro` für den Intel Pentium Pro.
- `Pentium-II` für den Intel Pentium II oder den Pre-Coppermine Celeron.
- `Pentium-III` für den Intel Pentium III oder Coppermine Celeron.
- `Pentium-4` für den Intel Pentium 4 oder den P4-basierten Celeron.
- `K6` für den AMD K6, K6-II und K6-III (auch bekannt als K6-3D).
- `Athlon` für die AMD K7-Familie (Athlon/Duron/Thunderbird).
- `Opteron/Athlon64/Hammer/K8` für alle K8 und neuere AMD-CPUs.
- `Crusoe` für die Transmeta Crusoe-Serie.
- `Efficeon` für die Transmeta Efficeon-Reihe.
- `Winchip-C6` für den ursprünglichen IDT-Winchip.
- `Winchip-2` für IDT-Winchips mit 3dNow! Fähigkeiten.
- `AMD Elan` für die 32-Bit AMD Elan Embedded CPU.
- `GeodeGX1` für Geode GX1 (Cyrix MediaGX).
- `Geode GX/LX` für AMD Geode GX und LX Prozessoren.
- `CyrixIII/VIA C3` für VIA Cyrix III oder VIA C3.
- `VIA C3-2` für VIA C3-2 "Nehemiah"(Modell 9 und höher).
- `VIA C7` für VIA C7.
- `Intel P4` für die Pentium 4/Netburst-Mikroarchitektur.
- `Core 2/newer Xeon` für alle Core2 und neueren Intel-CPUs.
- `Intel Atom` für die CPUs mit Atom-Mikroarchitektur.
- `Generic-x86-64` für einen Kernel, der auf jeder x86-64-CPU läuft.

Weitere Details finden Sie im Hilfetext der jeweiligen Option. Wenn Sie nicht wissen, was Sie tun sollen, wählen Sie `486`.

Derzeit (Kernelversion 6.6.x) können Sie nur aus fünf auswählen:

3.11.1 Opteron/Athlon64/Hammer/K8

CONFIG_MK8 [=n] **[N]**
Wählen Sie diese Option für einen Prozessor der AMD Opteron- oder Athlon64 Hammer-Familie. Ermöglicht die Verwendung einiger erweiterter Anweisungen und übergibt entsprechende Optimierungsflags an den GCC.

3.11.2 Intel P4 / older Netburst based Xeon

CONFIG_MPSC [=n] **[N]**
Optimiert für Intel Pentium 4, Pentium D und ältere Nocona/Dempsey Xeon CPUs mit Intel 64bit, die mit x86-64 kompatibel sind. Beachten Sie, dass die neuesten Xeons (Xeon 51xx und 53xx) nicht auf dem Netburst-Kern basieren und diese Option nicht verwenden sollten.
Sie können sie anhand des Feldes cpu family in /proc/cpuinfo unterscheiden. Familie 15 ist ein älterer Xeon, Familie 6 ein neuerer.

3.11.3 Intel P4 / older Netburst based Xeon

CONFIG_MCORE2 [=n] **[Y]**
Wählen Sie dies für Intel Core 2 und neuere Core 2 Xeons (Xeon 51xx und 53xx) CPUs.
Sie können neuere von älteren Xeons anhand der CPU-Familie in /proc/cpuinfo unterscheiden. Neuere haben 6 und ältere 15 (kein Tippfehler).

3.11.4 Intel Atom

CONFIG_MATOM [=n] **[N]**
Wählen Sie diese Option für die Intel Atom-Plattform. Intel Atom CPUs haben eine In-Order-Pipelining-Architektur und können daher von entsprechend optimiertem Code profitieren. Verwenden Sie einen aktuellen GCC mit spezieller Atom-Unterstützung, um die Vorteile dieser Option voll ausschöpfen zu können.

3.11.5 Generic-x86-64

CONFIG_GENERIC_CPU [=y] **[N]**
Allgemeine x86-64-CPU. Läuft gleich gut auf allen x86-64-CPUs.

3.12 Old AMD GART IOMMU support

CONFIG_GART_IOMMU [=n] **[N]**
Bietet einen Treiber für ältere AMD Athlon64/Opteron/Turion/Sempron GART basierte Hardware IOMMUs an. Der GART unterstützt vollen DMA-Zugriff für Geräte mit 32-Bit-Zugriffsbeschränkungen auf Systemen mit mehr als 3 GB. Dies wird normalerweise für USB, Sound, viele IDE/SATA-Chipsätze und einige andere Geräte benötigt. Neuere Systeme haben in der Regel eine moderne AMD IOMMU, die über die Konfigurationsoption CONFIG_AMD_IOMMU=y unterstützt wird. In normalen Konfigurationen ist dieser Treiber nur aktiv, wenn er benötigt wird: Es sind mehr als 3 GB Arbeitsspeicher vorhanden und das System enthält ein auf 32 Bit begrenztes Gerät.
Wenn Sie unsicher sind, sagen Sie Y.

3.13 Enable Maximum number of SMP Processors and NUMA Nodes

CONFIG_MAXSMP [=n] **[N]**
Aktivieren der maximalen Anzahl von CPUs- und NUMA-Knoten für diese Architektur.
Wenn Sie unsicher sind, sagen Sie N.

3.14 Maximum number of CPUs

CONFIG_NR_CPUS [=320] **[8]**
Hier können Sie die maximale Anzahl von CPUs angeben, die dieser Kernel unterstützen soll. Wenn CPUMASK_OFFSTACK aktiviert ist, ist der maximal unterstützte Wert 8192, andernfalls ist der maximale Wert 512. Der Mindestwert, der Sinn macht, ist 2.

Dies dient lediglich dazu, Speicher zu sparen: jede unterstützte CPU fügt dem Kernel-Image etwa 8 kB hinzu.

3.15 Cluster scheduler support

CONFIG_SCHED_CLUSTER [=y] **[N]**
Die Unterstützung des Cluster-Schedulers verbessert die Entscheidungsfindung des CPU-Schedulers beim Umgang mit Maschinen, die Cluster von CPUs haben. Mit Cluster sind in der Regel mehrere CPUs gemeint, die eng beieinander liegen und sich Mid-Level-Caches, Last-Level-Cache-Tags oder interne Busse teilen.

3.16 Multi-core scheduler support

CONFIG_SCHED_MC [=y] **[Y]**
Die Unterstützung des Multi-Core-Schedulers verbessert die Entscheidungsfindung des CPU-Schedulers beim Umgang mit Multi-Core-CPU-Chips auf Kosten eines leicht erhöhten Overheads an einigen Stellen. Wenn Sie unsicher sind, geben Sie hier N an.

3.16.1 CPU core priorities scheduler support

CONFIG_SCHED_MC_PRIO [=y] **[Y]**
Bei CPUs mit Intel Turbo-Boost-Max-Technik 3.0 wird die Reihenfolge der Kerne bei der Herstellung festgelegt, so dass bestimmte Kerne höhere Turbofrequenzen erreichen können (bei Single-Thread-Arbeitslasten) als andere. Durch die Aktivierung dieser Kernel-Funktion wird der Scheduler über die TBM3- (auch ITMT-) Prioritätsreihenfolge der CPU-Kerne informiert und passt die CPU-Auswahllogik des Schedulers entsprechend an, so dass eine höhere Gesamtsystemleistung erzielt werden kann. Diese Funktion hat keine Auswirkungen auf CPUs ohne diese Funktion.
Wenn Sie unsicher sind, geben Sie hier Y an.

3.17 Reroute for broken boot IRQs

CONFIG_X86_REROUTE_FOR_BROKEN_BOOT_IRQS [=y] **[Y]**
Diese Option ermöglicht eine Umgehung, die eine Quelle für unerwünschte Unterbrechungen behebt. Dies wird empfohlen, wenn die Thread-Interrupt-Behandlung auf Systemen verwendet wird, bei denen die Erzeugung von überflüssigen „Boot-Interrupts“ nicht deaktiviert werden kann. Einige Chipsätze erzeugen einen Legacy-INTx-„Boot-IRQ“, wenn der IRQ-Eintrag im IO-APIC des Chipsatzes maskiert ist (wie es z. B. der RT-Kernel während der Interruptbehandlung tut). Bei Chipsätzen, bei denen diese Boot-IRQ-Erzeugung nicht deaktiviert werden kann, wird durch diese Abhilfe die ursprüngliche IRQ-Leitung maskiert, so dass nur der entsprechende „Boot-IRQ“ an die CPUs geliefert wird. Die Problemumgehung weist den Kernel außerdem an, den IRQ-Handler auf der Boot-IRQ-Leitung einzurichten. Auf diese Weise wird nur ein Interrupt an den Kernel geliefert. Andernfalls kann der zweite Interrupt den Kernel dazu veranlassen, (lebenswichtige) Interrupt-Leitungen herunterzufahren. Betrifft nur „defekte“ Chipsätze. Die gemeinsame Nutzung von Interrupts kann auf diesen Systemen erhöht werden.

3.18 Machine Check / overheating reporting

CONFIG_X86_MCE [=y] **[Y]**
(Maschinenprüfung / Überhitzungsmeldung) Durch die Unterstützung von Machine Check kann der Prozessor den Kernel benachrichtigen, wenn er ein Problem feststellt (z. B. Überhitzung, Datenbeschädigung). Welche Maßnahmen der Kernel ergreift, hängt von der Schwere des Problems ab und reicht von Warnmeldungen bis zum Anhalten des Rechners.

3.18.1 Support for deprecated /dev/mcelog character device

CONFIG_X86_MCELOG_LEGACY [=n] **[N]**
Aktivierung der Unterstützung für /dev/mcelog, die vom alten mcelog-Benutzerraum-Logging-Daemon (mcelog userspace logging daemon) benötigt wird. Erwägen Sie den Umstieg auf die neue Generation des rasdaemon.

3.18.2 Intel MCE features

CONFIG_X86_MCE_INTEL [=y] **[Y]**
Zusätzliche Unterstützung für Intel-spezifische MCE-Funktionen wie den Temperaturmonitor (thermal monitor).

3.18.3 AMD MCE features

CONFIG_X86_MCE_AMD [=y] **[N]**
Zusätzliche Unterstützung für AMD-spezifische MCE-Funktionen wie den DRAM-Fehlerschwellenwert (DRAM Error Threshold).

3.19 Machine check injector support

CONFIG_X86_MCE_INJECT [=m] **[M]**
Unterstützung bei der Einspeisung von Maschinenprüfungen zu Testzwecken. Wenn Sie nicht wissen, was eine Maschinenprüfung ist und Sie keine Kernel-Qualitätssicherung durchführen, können Sie mit Sicherheit N sagen, also nein.

3.20 Performance monitoring →

Leistungsüberwachung

3.20.1 Intel uncore performance events

CONFIG_PERF_EVENTS_INTEL_UNCORE [=m] **[M]**
Unterstützung für Intel-Uncore-Leistungsereignisse. Diese sind auf NehalemEX und moderneren Prozessoren verfügbar.

3.20.2 Intel/AMD rapl performance events

CONFIG_PERF_EVENTS_INTEL_RAPL [=m] **[M]**
Unterstützung für Intel- und AMD-RAPL-Leistungsereignisse zur Leistungsüberwachung auf modernen Prozessoren.

3.20.3 Intel cstate performance events

CONFIG_PERF_EVENTS_INTEL_CSTATE [=m] **[M]**
Einbeziehung der Unterstützung für Intel cstate performance events für die Leistungsüberwachung auf modernen Prozessoren.

3.20.4 AMD Processor Power Reporting Mechanism

CONFIG_PERF_EVENTS_AMD_POWER [=m] **[M]**
Unterstützung von Stromversorgungsberichten für AMD-Prozessoren.
Derzeit wird die Schnittstelle X86_FEATURE_ACC_POWER (CPUID Fn8000_0007_EDX[12]) genutzt, um den durchschnittlichen Stromverbrauch von Prozessoren der Familie 15h zu berechnen.

3.20.5 AMD Uncore performance events

CONFIG_PERF_EVENTS_AMD_UNCORE [=m] **[M]**
Unterstützung für AMD-Uncore-Leistungsereignisse für die Verwendung mit z.B.
`perf stat -e amd_l3/.../,amd_df/.../`.
Um diesen Treiber als Modul zu kompilieren, wählen Sie hier M: das Modul wird `amd-uncore` genannt.

3.20.6 AMD Zen3 Branch Sampling support

CONFIG_PERF_EVENTS_AMD_BRS [=y] **[Y]**
Aktivieren Sie die AMD Zen3 Branch Sampling-Unterstützung (BRS), die bis zu 16 aufeinanderfolgende Verzweigungen in Registern erfasst.

3.20.7 IOPERM and IOPL Emulation

CONFIG_X86_IOPL_IOPERM [=y] **[Y]**
Dies ermöglicht die ioperm()- und iopl()-Systemaufrufe, die für Legacy-Anwendungen erforderlich sind. Bei der Legacy-IOPL-Unterstützung handelt es sich um einen weitreichenden Mechanismus, der es dem Userspace ermöglicht, neben dem Zugriff auf alle 65536 E/A-Ports auch Interrupts zu deaktivieren. Um diesen Zugriff zu erhalten, benötigt der Aufrufer CAP_SYS_RAWIO-Fähigkeiten und die Erlaubnis von potenziell aktiven Sicherheitsmodulen. Die Emulation schränkt die Funktionalität des Syscalls auf den Zugriff auf alle E/A-Ports ein, verhindert aber die Möglichkeit, Interrupts aus dem Userspace zu deaktivieren, was bei Verwendung des Hardware-IOPL-Mechanismus möglich wäre.

3.20.8 Late microcode loading (DANGEROUS)

CONFIG_MICROCODE_LATE_LOADING [=n] **[N]**
Das späte Laden von Mikrocode, wenn das System bereits läuft und Befehle ausführt, ist eine heikle Angelegenheit und sollte nach Möglichkeit vermieden werden. Allein die Abfolge der Synchronisierung aller Kerne und SMT-Threads ist ein zerbrechlicher Tanz, der nicht garantiert, dass die Kerne nach dem Laden nicht softlocking werden. Daher sollten Sie dies auf eigenes Risiko tun. Das späte Laden färbt auch den Kernel.

3.20.9 /dev/cpu/*/msr - Model-specific register support

CONFIG_X86_MSR [=y] **[Y]**
Dieses Gerät ermöglicht privilegierten Prozessen den Zugriff auf die modellspezifischen x86-Register (MSRs).
Es ist ein Zeichengerät mit Major 202 und Minors 0 bis 31 für `/dev/cpu/0/msr` bis `/dev/cpu/31/msr`. MSR-Zugriffe sind bei Multiprozessorsystemen an eine bestimmte CPU gerichtet.

3.20.10 /dev/cpu/*/cpuid - CPU information support

CONFIG_X86_CPUID [=y] **[Y]**
Dieses Gerät ermöglicht Prozessen den Zugriff auf den x86 CPUID-Befehl, der auf einem bestimmten Prozessor ausgeführt werden soll. Es handelt sich um ein Zeichengerät mit Major 203 und Minors 0 bis 31 für `/dev/cpu/0/cpuid` bis `/dev/cpu/31/cpuid`.

3.20.11 Enable 5-level page tables support

CONFIG_X86_5LEVEL [=y] **[Y]**
5-Level-Paging ermöglicht den Zugriff auf einen größeren Adressraum: bis zu 128 PiB virtueller Adressraum und 4 PiB physikalischer Adressraum. Es wird von zukünftigen Intel-CPUs unterstützt werden. Ein Kernel mit aktivierter Option kann auf Rechnern gebootet werden, die 4- oder 5-Level-Paging unterstützen. Siehe Documentation/arch/x86/x86_64/5level-paging.rst für weitere Informationen.
Sagen Sie N, wenn Sie unsicher sind.

3.20.12 Enable statistic for Change Page Attribute

CONFIG_X86_CPA_STATISTICS [=y] **[Y]**
Statistiken über den Mechanismus zum Ändern von Seitenattributen offenlegen, der dabei hilft, die Wirksamkeit der Erhaltung großer und umfangreicher Seitenzuordnungen zu bestimmen, wenn Zuordnungsschutzmaßnahmen geändert werden.

3.20.13 AMD Secure Memory Encryption (SME) support

CONFIG_AMD_MEM_ENCRYPT [=y] **[N]**
Sagen Sie Ja, um die Unterstützung für die Verschlüsselung des Systemspeichers zu aktivieren. Dies erfordert einen AMD-Prozessor, der Secure Memory Encryption (SME) unterstützt.

3.20.13.1 Activate AMD Secure Memory Encryption (SME) by default

CONFIG_AMD_MEM_ENCRYPT_ACTIVE_BY_DEFAULT [=n] **[N]**
Sagen Sie Ja, damit der Systemspeicher standardmäßig verschlüsselt wird, wenn er auf einem AMD-Prozessor läuft, der Secure Memory Encryption (SME) unterstützt. Wenn Sie Y wählen, kann die Verschlüsselung des Systemspeichers mit der Befehlszeilenoption `mem_encrypt=off` deaktiviert werden. Ist der Wert auf N gesetzt, kann die Verschlüsselung des Systemspeichers mit der Befehlszeilenoption `mem_encrypt=on` aktiviert werden.

3.20.14 NUMA Memory Allocation and Scheduler Support

CONFIG_NUMA [=y] **[Y]**
Aktivieren Sie die NUMA-Unterstützung (Non-Uniform Memory Access). Der Kernel wird versuchen, den von einer CPU verwendeten Speicher dem lokalen Speicher-Controller der CPU zuzuweisen und dem Kernel mehr Kenntnis über NUMA zu geben.
Für 64-Bit wird dies empfohlen, wenn das System Intel Core i7 (oder höher), AMD Opteron oder EM64T NUMA ist.
Für 32-Bit ist dies nur erforderlich, wenn Sie einen 32-Bit-Kernel auf einer 64-Bit-NUMA-Plattform booten. Andernfalls sollten Sie N angeben.

3.20.14.1 Old style AMD Opteron NUMA detection

CONFIG_AMD_NUMA [=y] **[N]**
Aktivieren Sie die Erkennung der AMD NUMA-Knoten-Topologie. Wenn Sie ein AMD-Multiprozessorsystem haben, sollten Sie hier Y angeben. Dies verwendet eine alte Methode, um die NUMA-Konfiguration direkt von der eingebauten Northbridge des Opteron zu lesen.
Es wird empfohlen, stattdessen X86_64_ACPI_NUMA zu verwenden, das auch Priorität hat, wenn beide einkompiliert sind.

3.20.14.2 ACPI NUMA detection

CONFIG_X86_64_ACOU_NUMA [=y] **[Y]**
Aktivieren Sie die auf ACPI SRAT basierende Knoten-Topologie-Erkennung.

3.20.14.3 NUMA emulation

CONFIG_NUMA_EMU [=n] **[N]**
Aktivieren Sie die NUMA-Emulation. Eine flache Maschine wird in virtuelle Knoten aufgeteilt, wenn sie mit `numa=fake=N` gebootet wird, wobei N die Anzahl der Knoten ist. Dies ist nur für die Fehlersuche nützlich.

3.20.14.4 Maximum NUMA Nodes (as a power of 2)

CONFIG_NODES_SHIFT [=5] **[5]**
(Maximale NUMA-Knoten (als eine Potenz von 2))
Geben Sie die maximale Anzahl der auf dem Zielsystem verfügbaren NUMA-Knoten an. Erhöht den reservierten Speicherplatz für verschiedene Tabellen.

3.20.15 Enable sysfs memory/probe interface

CONFIG_ARCH_MEMORY_PROBE [=n] **[N]**
Diese Option aktiviert eine sysfs-Speicher/Probe-Schnittstelle für Tests.
Siehe Documentation/admin-guide/mm/memory-hotplug.rst für weitere Informationen. Wenn Sie unsicher sind, wie Sie diese Frage beantworten sollen, antworten Sie mit N.

3.20.16 Support non-standard NVDIMMs and ADR protected memory

CONFIG_X86_PMEM_LEGACY [=m] **[M]**
Behandeln Sie Speicher, der mit dem nicht standardmäßigen e820-Typ von 12 markiert ist, wie er vom Intel Sandy Bridge-EP Referenz-BIOS verwendet wird, als geschützten Speicher. Der Kernel bietet diese Regionen dem `pmem`-Treiber an, so dass sie für persistenten Speicher verwendet werden können.
Sagen Sie Y, wenn Sie unsicher sind.

3.20.17 Check for low memory corruption

CONFIG_X86_CHECK_BIOS_CORRUPTION [=y] **[Y]**
Regelmäßige Überprüfung auf Speicherbeschädigung im niedrigen Speicher, die vermutlich durch das BIOS verursacht wird. Auch wenn dies in der Konfiguration aktiviert ist, zur Laufzeit ist es deaktiviert. Aktivieren Sie es, indem Sie `memory_corruption_check=1` in der Kernel-Befehlszeile eingeben. Standardmäßig werden die unteren 64 k des Speichers alle 60 Sekunden überprüft; siehe die Parameter `memory_corruption_check_size` und `memory_corruption_check_period` in Documentation/admin-guide/kernel-parameters.rst, um dies anzupassen. Wenn diese Option mit den Standardparametern aktiviert ist, hat sie so gut wie keinen Overhead, da sie eine relativ kleine Menge an Speicher reserviert und diesen nur selten durchsucht. Sie erkennt Korruption und verhindert, dass sie das laufende System beeinträchtigt. Sie ist jedoch als Diagnosewerkzeug gedacht; wenn eine wiederholte BIOS-verursachte Beschädigung stets denselben Speicher betrifft, können Sie `memmap=` verwenden, um zu verhindern, dass der Kernel diesen Speicher verwendet.

Hinweis: Kann ausgeschaltet werden, wenn im `journalctl` niemals „corrupted low memory“ erscheint.

3.20.17.1 Set the default setting of memory_corruption_check

CONFIG_X86_BOOTPARAM_MEMORY_CORRUPTION_CHECK [=y] **[Y]**
Legt fest, ob der Standardstatus von `memory_corruption_check` ein- oder ausgeschaltet ist.

3.20.18 MTRR (Memory Type Range Register) support

CONFIG_MTRR [=y] **[Y]**
Bei Prozessoren der Intel P6-Familie (Pentium Pro, Pentium II und später) können die Memory Type Range Register (MTRRs) verwendet werden, um den Zugriff des Prozessors auf Speicherbereiche zu steuern. Dies ist besonders nützlich, wenn Sie eine Videokarte (VGA) an einem PCI- oder AGP-Bus haben. Durch die Aktivierung von Write-Combining können Bus-Schreibübertragungen zu einer größeren Übertragung kombiniert werden, bevor sie über den PCI/AGP-Bus geleitet werden. Dies kann die Leistung von Bildschreiboperationen um das 2,5-fache oder mehr erhöhen. Wenn Sie hier Y angeben, wird eine /proc/mtrr-Datei erstellt, die zur Manipulation der MTRRs Ihres Prozessors verwendet werden kann. Normalerweise sollte der X-Server dies verwenden.
Dieser Code hat eine recht generische Schnittstelle, so dass ähnliche Steuerregister auf anderen Prozessoren ebenfalls leicht unterstützt werden können:
Die Prozessoren Cyrix 6x86, 6x86MX und M II verfügen über Address Range Registers (ARRs), die eine ähnliche Funktionalität wie MTRRs bieten. In diesen Fällen werden die ARRs zur Emulation der MTRRs verwendet. Die AMD-Prozessoren K6-2 (Stepping 8 und höher) und K6-3 haben zwei MTRRs. Der Centaur C6 (WinChip) hat 8 MCRs, die Schreibkombinationen ermöglichen. Alle diese Prozessoren werden von diesem Code unterstützt und es ist sinnvoll, hier Y anzugeben, wenn Sie einen dieser Prozessoren haben.
Die Angabe von Y an dieser Stelle behebt auch ein Problem mit fehlerhaften SMP-BIOSen, die die MTRRs nur für die Boot-CPU und nicht für die sekundären CPUs setzen. Das kann zu allen möglichen Problemen führen, also ist es gut, hier Y zu sagen.
Sie können sicher Y sagen, auch wenn Ihr Rechner keine MTRRs hat, Sie werden nur etwa 9 KB zu Ihrem Kernel hinzufügen. Siehe <file:Documentation/arch/x86/mtrr.rst> für weitere Informationen.

3.20.18.1 MTRR cleanup support

CONFIG_MTRR_SANITIZER [=y] **[Y]**
Umwandlung des MTRR-Layouts von kontinuierlich in diskret, damit X-Treiber Rückschreibeinträge hinzufügen können. Kann mit `disable_mtrr_cleanup` in der Kernel-Kommandozeile deaktiviert werden. Die größte MTRR-Eintragsgröße für einen kontinuierlichen Block kann mit `mtrr_chunk_size` festgelegt werden.
Wenn Sie unsicher sind, sagen Sie Y.

3.20.18.2 MTRR cleanup enable value (0-1)

CONFIG_MTRR_SANITIZER [=1] **[1]**
Aktivieren Sie den „mtrr cleanup“-Standardwert

3.20.18.3 MTRR cleanup spare reg num (0-7)

CONFIG_MTRR_SANITIZER_SPARE_REG_NR_DEFAULT [=0] **[0]**
MTRR cleanup spare entries Defaulteintrag, dies kann über `mtrr_spare_reg_nr=N` auf der Kernel-Befehlszeile geändert werden.

3.20.19 Indirect Branch Tracking

CONFIG_X86_KERNEL_IBT [=y] **[Y]**
Bauen Sie den Kernel mit Unterstützung für Indirect Branch Tracking auf, eine Hardware-Unterstützung, die die Integrität des Kontrollflusses an den Rändern schützt. Sie erzwingt, dass alle indirekten Aufrufe auf einer ENDBR-Anweisung landen müssen, und der Compiler wird den Code mit ihnen instrumentieren, damit dies geschieht.
Zusätzlich zur Erstellung des Kernels mit IBT werden alle Funktionen, die keine indirekten Aufrufziele sind, versiegelt, um zu verhindern, dass sie jemals zu solchen werden.
Dies erfordert LTO wie objtool-Läufe und verlangsamt den Bau. Es reduziert jedoch die Anzahl der ENDBR-Anweisungen im Kernel-Image erheblich.

3.20.20 Memory Protection Keys

CONFIG_X86_INTEL_MEMORY_PROTECTION_KEYS [=y] **[Y]**
Memory Protection Keys bietet einen Mechanismus zur Erzwingung seitenbasierter Schutzmaßnahmen, ohne dass die Seitentabellen geändert werden müssen, wenn eine Anwendung ihre Schutzdomänen ändert.
Einzelheiten siehe Documentation/core-api/protection-keys.rst
Wenn Sie unsicher sind, sagen Sie Y.

3.20.21 TSX enable mode () →

CONFIG_X86_INTEL_MEMORY_PROTECTION_KEYS [=y] **[Y]**
Intels TSX-Funktion (Transactional Synchronization Extensions) ermöglicht die Optimierung von Sperrprotokollen durch Lock Elision, was zu einer spürbaren Leistungssteigerung führen kann. Andererseits hat sich gezeigt, dass TSX für Seitenkanalangriffe (z. B. TAA) ausgenutzt werden kann, und es ist wahrscheinlich, dass in Zukunft weitere Angriffe dieser Art entdeckt werden. Daher ist TSX standardmäßig nicht aktiviert (aka `tsx=off`). Ein Administrator kann diese Entscheidung durch den Befehlszeilenparameter `tsx=on` außer Kraft setzen. Auch wenn TSX aktiviert ist, versucht der Kernel, die bestmögliche TAA-Abschwächung zu aktivieren, je nach dem für den jeweiligen Rechner verfügbaren Mikrocode. Mit dieser Option kann der Standard-Tsx-Modus zwischen `tsx=on`, `=off` und `=auto` eingestellt werden. Siehe Documentation/admin-guide/kernel-parameters.txt für weitere Details. Sagen Sie off, wenn Sie sich nicht sicher sind, auto, wenn TSX in Gebrauch ist, aber auf sicheren Plattformen verwendet werden sollte, oder on, wenn TSX in Gebrauch ist und der Sicherheitsaspekt von tsx nicht relevant ist.

3.20.21.1 off

CONFIG_X86_INTEL_TSX_MODE_OFF [=n] **[N]**
TSX ist, wenn möglich, deaktiviert – entspricht dem Befehlszeilenparameter `tsx=off`.

3.20.21.2 on

CONFIG_X86_INTEL_TSX_MODE_ON [=n] **[N]**
TSX ist auf TSX-fähiger Hardware immer aktiviert – gleichbedeutend mit dem Befehlszeilenparameter `tsx=on`

3.20.21.3 auto

CONFIG_X86_INTEL_TSX_MODE_AUTO [=y] **[Y]**
TSX wird auf TSX-fähiger Hardware aktiviert, die als sicher gegen Seitenkanalangriffe gilt – gleichbedeutend mit dem Befehlszeilenparameter `tsx=auto`.

3.20.22 Software Guard eXtensions (SGX)

CONFIG_X86_SGX [=y] **[Y]**
Intel(R) Software Guard eXtensions (SGX) ist eine Reihe von CPU-Befehlen, die von Anwendungen verwendet werden können, um private Code- und Datenbereiche, die so genannten Enklaven, zu reservieren.

Auf den privaten Speicher einer Enklave kann nur von Code zugegriffen werden, der innerhalb der Enklave läuft. Zugriffe von außerhalb der Enklave, einschließlich anderer Enklaven, werden von der Hardware nicht zugelassen.
Wenn Sie unsicher sind, sagen Sie N.

3.20.23 X86 userspace shadow stack

CONFIG_X86_USER_SHADOW_STACK [=y] **[Y]**
Der Schattenstapelschutz ist eine Hardwarefunktion, die eine Beschädigung der Rücksprungadresse einer Funktion erkennt. Dies hilft, ROP-Angriffe abzuschwächen. Anwendungen müssen aktiviert sein, um sie zu nutzen, und der alte Userspace erhält den Schutz nicht ümsonst". CPUs, die Shadow Stacks unterstützen, wurden erstmals im Jahr 2020 vorgestellt. Weitere Informationen finden Sie unter Documentation/arch/x86/shstk.rst.
Wenn Sie unsicher sind, sagen Sie N.

3.20.24 EFI runtime service support

CONFIG_EFI [=y] **[Y]**
Dies ermöglicht es dem Kernel, verfügbare EFI-Laufzeitdienste (wie die EFI-Variablendienste) zu nutzen. Diese Option ist nur auf Systemen mit EFI-Firmware sinnvoll. Außerdem sollten Sie den neuesten ELILO-Lader verwenden, der unter `http://elilo.sourceforge.net` verfügbar ist, um die Vorteile der EFI-Laufzeitdienste zu nutzen. Aber auch mit dieser Option sollte der resultierende Kernel weiterhin auf bestehenden Nicht-EFI-Plattformen booten.

3.20.24.1 EFI stub support

CONFIG_EFI_STUB [=y] **[Y]**
Mit dieser Kernel-Funktion kann ein bzImage direkt von der EFI-Firmware geladen werden, ohne dass ein Bootloader erforderlich ist.
Weitere Informationen finden Sie unter Documentation/admin-guide/efi-stub.rst.

3.20.24.1.1 EFI handover protocol (DEPRECATED)

CONFIG_EFI_STUB [=y] **[Y]**
(EFI-Übergabeprotokoll (VERALTET))
Wählen Sie dies, um Unterstützung für das veraltete EFI-Handover-Protokoll zu erhalten, das alternative Einstiegspunkte in den EFI-Stub definiert. Dies ist eine Praxis, die keine Grundlage in der UEFI-Spezifikation hat und ein Vorwissen seitens des Bootloaders über Linux/x86-spezifische Wege der Übergabe der Kommandozeile und initrd erfordert, und wo im Speicher diese Assets geladen werden können.
Im Zweifelsfall sagen Sie Y. Auch wenn die entsprechende Unterstützung im Upstream-GRUB oder anderen Bootloadern nicht vorhanden ist, bauen die meisten Distros GRUB mit zahlreichen Downstream-Patches und können sich daher auf das Handover-Protokoll verlassen.

3.20.24.1.2 EFI mixed-mode support

CONFIG_EFI_MIXED [=y] **[Y]**
Wenn Sie diese Funktion aktivieren, kann ein 64-Bit-Kernel auf einer 32-Bit-Firmware gebootet werden, vorausgesetzt, Ihre CPU unterstützt den 64-Bit-Modus.
Beachten Sie, dass es nicht möglich ist, einen Mixed-Mode-fähigen Kernel über den EFI-Boot-Stub zu booten – es muss ein Bootloader verwendet werden, der das EFI-Handover-Protokoll unterstützt.
Wenn Sie unsicher sind, sagen Sie N.

3.20.24.2 Enable EFI fake memory map

CONFIG_EFI_FAKE_MEMMAP [=n] **[N]**
Wenn Sie hier Y angeben, wird die Boot-Option `efi_fake_mem` aktiviert. Durch Angabe dieses Parameters können Sie einem bestimmten Speicherbereich beliebige Attribute hinzufügen, indem Sie die ursprüngliche (von der Firmware bereitgestellte) EFI-Memmap aktualisieren. Dies ist nützlich für das Debugging von EFI-Memmap-bezogenen Funktionen, z.B. Address Range Mirroring.

3.20.25 Timer frequency () →

Ermöglicht die Konfiguration der Timer-Frequenz. Es ist üblich, den Timer-Interrupt mit 1000 Hz laufen zu lassen, aber 100 Hz kann für Server und NUMA-Systeme vorteilhafter sein, die keine schnelle Reaktion für die Benutzerinteraktion benötigen und bei denen es zu Buskonflikten und Cacheline-Bounches als Folge von Timer-Interrupts kommen kann. Beachten Sie, dass der Timer-Interrupt in einer SMP-Umgebung auf jedem Prozessor auftritt, was zu NR_CPUS * HZ Anzahl der Timer-Interrupts pro Sekunde führt.

3.20.25.1 100 Hz

CONFIG_HZ_100 [=n] **[N]**
100 Hz ist eine typische Wahl für Server, SMP- und NUMA-Systeme mit vielen Prozessoren, die eine geringere Leistung aufweisen können, wenn zu viele Timer-Interrupts auftreten.

3.20.25.2 250 Hz

CONFIG_HZ_250 [=n] **[N]**
250 Hz ist ein guter Kompromiss, der eine gute Serverleistung ermöglicht und auch auf SMP- und NUMA-Systemen eine gute interaktive Reaktionsfähigkeit zeigt. Wenn Sie NTSC-Video oder Multimedia verwenden, wählen Sie stattdessen 300 Hz.

3.20.25.3 300 Hz

CONFIG_HZ_300 [=y] **[Y]**
300 Hz ist ein guter Kompromiss, der eine gute Serverleistung und gleichzeitig eine gute interaktive Reaktionsfähigkeit selbst auf SMP- und NUMA-Systemen ermöglicht und sowohl bei PAL- als auch bei NTSC-Bildraten für Video- und Multimedia-Arbeiten genau eingehalten wird.

3.20.25.4 1000 Hz

CONFIG_HZ_1000 [=n] **[N]**
1000 Hz ist die bevorzugte Wahl für Desktop-Systeme und andere Systeme, die schnelle interaktive Reaktionen auf Ereignisse erfordern.

3.20.26 Physical address where the kernel is loaded

CONFIG_PHYSICAL_START [=0x1000000] **[0x1000000]**
Dies gibt die physikalische Adresse an, unter der der Kernel geladen wird. Wenn der Kernel nicht verschiebbar ist (CONFIG_RELOCATABLE=n), dekomprimiert sich bzImage an die oben genannte physikalische Adresse und wird von dort aus gestartet. Andernfalls wird bzImage von der Adresse aus gestartet, an der es vom Bootloader geladen wurde, und ignoriert die obige physikalische Adresse. In normalen kdump-Fällen muss diese Option nicht gesetzt/geändert werden, da bzImage nun als vollständig relozierbares Image (CONFIG_RELOCATABLE=y) kompiliert und zum Laden und Ausführen von einer anderen Adresse verwendet werden kann. Diese Option ist vor allem für die Leute nützlich, die kein bzImage für die Erfassung des Crash-Dumps verwenden wollen und stattdessen vmlinux einsetzen wollen. vmlinux ist nicht relocatable, daher muss ein Kernel speziell kompiliert werden, um von einem bestimmten Speicherbereich (normalerweise ein reservierter Bereich) zu laufen, und diese Option ist sehr nützlich. Wenn Sie also bzImage zum Erfassen des Crash-Dumps verwenden, lassen Sie den Wert hier unverändert auf 0x1000000 und setzen Sie CONFIG_RELOCATABLE=y.
Andernfalls, wenn Sie vmlinux für die Aufzeichnung des Crash-Dumps verwenden wollen, ändern Sie diesen Wert auf den Beginn des reservierten Bereichs. Mit anderen Worten, er kann auf der Grundlage des "XWertes gesetzt werden, wie er im "crashkernel=YM@XMBefehlszeilen-Boot-Parameter angegeben ist, der an den panic-ed-Kernel übergeben wird. Weitere Details zu Crash Dumps finden Sie in Documentation/admin-guide/kdump/kdump.rst. Die Verwendung von bzImage für die Aufzeichnung des Crash-Dumps wird empfohlen, da man nicht zwei Kernel erstellen muss. Derselbe Kernel kann als Produktionskernel und als Erfassungskernel verwendet werden. Die obige Option sollte verschwinden, nachdem die Unterstützung von relocatable bzImage eingeführt wurde. Sie ist aber noch vorhanden, weil es Benutzer gibt, die weiterhin vmlinux für die Dump-Erfassung verwenden. Diese Option sollte im Laufe der Zeit verschwinden. Ändern Sie diese Option nicht, wenn Sie nicht wissen, was Sie tun.

3.20.27 Build a relocatable kernel

CONFIG_RELOCATABLE [=y] **[Y]**
Dadurch wird ein Kernel-Image erstellt, das die Informationen über den Standortwechsel beibehält, so dass es an einem anderen Ort als den standardmäßigen 1 MB geladen werden kann. Die Verschiebungen machen das Kernel-Binary etwa 10 % größer, werden aber zur Laufzeit verworfen. Eine Anwendung ist der kexec on panic-Fall, bei dem der Wiederherstellungs-Kernel an einer anderen physikalischen Adresse liegen muss als der primäre Kernel. Anmerkung: Wenn CONFIG_RELOCATABLE=y ist, dann läuft der Kernel von der Adresse aus, an der er geladen wurde, und von der zur Kompilierzeit physische Adresse (CONFIG_PHYSICAL_START) als Mindeststandort verwendet.

3.20.27.1 Randomize the address of the kernel image (KASLR)

CONFIG_RANDOMIZE_BASE [=y] **[Y]**
Zur Unterstützung der Kernel Address Space Layout Randomization (KASLR) werden die physische Adresse, an der das Kernel-Image dekomprimiert wird, und die virtuelle Adresse, auf die das Kernel-Image abgebildet wird, randomisiert. Dies ist ein Sicherheitsmerkmal, das Exploit-Versuche verhindert, die auf der Kenntnis des Speicherorts von Kernel-Code-Interna beruhen. Bei 64-Bit werden die physische und die virtuelle Adresse des Kernels getrennt randomisiert. Die physische Adresse liegt irgendwo zwischen 16 MB und dem Anfang des physischen Speichers (bis zu 64 TB). Die virtuelle Adresse wird von 16 MB bis zu 1 GB randomisiert (9 Bits Entropie). Beachten Sie, dass dadurch auch der für Kernel-Module verfügbare Speicherplatz von 1,5 GB auf 1 GB reduziert wird. Bei 32-Bit werden die physischen und virtuellen Adressen des Kernels zusammen randomisiert. Sie werden von 16 MB bis zu 512 MB randomisiert (8 Bits Entropie).
Die Entropie wird mit dem RDRAND-Befehl erzeugt, sofern er unterstützt wird. Wenn RDTSC unterstützt wird, wird sein Wert ebenfalls in den Entropie-Pool gemischt. Wenn weder RDRAND noch RDTSC unterstützt werden, wird die Entropie aus dem i8254-Zeitgeber gelesen. Die nutzbare Entropie ist dadurch begrenzt, dass der Kernel mit 2 GB-Adressierung aufgebaut ist und dass PHYSICAL_ALIGN mindestens 2 MB betragen muss. Infolgedessen sind theoretisch nur 10 Bits Entropie möglich, aber die Implementierungen sind aufgrund des Speicherlayouts noch weiter eingeschränkt.
Wenn Sie unsicher sind, sagen Sie Y.

3.20.28 Alignment value to which kernel should be aligned

CONFIG_PHYSICAL_ALIGN [=0x200000] **[0x200000]**
Dieser Wert legt die Ausrichtungsbeschränkungen für die physikalische Adresse fest, von der der Kernel geladen und ausgeführt wird. Der Kernel wird für eine Adresse kompiliert, die den obigen Ausrichtungsbeschränkungen entspricht. Wenn der Bootloader den Kernel an einer nicht ausgerichteten Adresse lädt und CONFIG_RELOCATABLE gesetzt ist, verschiebt sich der Kernel an die nächstgelegene Adresse, die auf den obigen Wert ausgerichtet ist, und wird von dort aus gestartet. Wenn der Bootloader den Kernel an einer nicht ausgerichteten Adresse lädt und CONFIG_RELOCATABLE nicht gesetzt ist, ignoriert der Kernel die Ladeadresse zur Laufzeit und dekomprimiert sich an die Adresse, für die er kompiliert wurde, und läuft von dort aus. Die Adresse, für die der Kernel kompiliert wurde, erfüllt bereits die oben genannten Ausrichtungsbeschränkungen. Das Endergebnis ist also, dass der Kernel von einer physikalischen Adresse aus läuft, die die oben genannten Ausrichtungsbeschränkungen erfüllt. Bei 32-Bit muss dieser Wert ein Vielfaches von 0x2000 sein. Bei 64-Bit muss dieser Wert ein Vielfaches von 0x200000 sein. Ändern Sie dies nicht, wenn Sie nicht wissen, was Sie tun.

3.20.29 Randomize the kernel memory sections

CONFIG_RANDOMIZE_MEMORY [=y] **[Y]**
Randomisiert die virtuelle Basisadresse von Kernel-Speicherabschnitten (physische Speicherzuordnung, vmalloc & vmemmap). Dieses Sicherheitsmerkmal macht Exploits, die sich auf vorhersehbare Speicherplätze verlassen, weniger zuverlässig. Die Reihenfolge der Zuweisungen bleibt unverändert. Entropie wird auf die gleiche Weise wie bei RANDOMIZE_BASE erzeugt. Aktuelle Implementierung in der optimalen Konfiguration haben im Durchschnitt 30.000 verschiedene mögliche virtuelle Adressen für jeden Speicherabschnitt.
Wenn Sie unsicher sind, sagen Sie Y.

3.20.30 Linear Address Masking support

CONFIG_ADDRESS_MASKING [=y] **[Y]**
Linear Address Masking (LAM) ändert die Prüfung, die auf lineare 64-Bit-Adressen angewandt wird, und ermöglicht der Software die nicht übersetzten Adressbits für Metadaten zu verwenden.
Diese Fähigkeit kann für die effiziente Implementierung von Adress-Sanitizern (ASAN) und für Optimierungen in JITs genutzt werden.

3.20.31 Disable the 32-bit vDSO (needed for glibc 2.3.3)

CONFIG_COMPAT_VDSO [=n] **[N]**
Bestimmte fehlerhafte Versionen der glibc stürzen ab, wenn sie mit einem 32-Bit vDSO konfrontiert werden, das nicht auf die in der Segmenttabelle angegebene Adresse abgebildet ist. Adresse zugeordnet ist, die in der Segmenttabelle angegeben ist.
Der Fehler wurde eingeführt durch f866314b89d56845f55e6f365e18b31ec978ec3a und behoben durch 3b3ddb4f7db98ec9e912ccdf54d35df4aa30e04a und 49ad572a70b8aeb91e57483a11dd1b77e31c4468.
Glibc 2.3.3 ist die einzige veröffentlichte Version mit dem Fehler, aber OpenSUSE 9 enthält eine fehlerhafte „glibc 2.3.2“. Das Symptom des Fehlers ist, dass alles beim Start abstürzt und sagt:
`dl_main: Assertion ` (void) ph->p_vaddr == _rtld_local._dl_sysinfo_dso`
Wenn Sie hier Y sagen, wird der Standardwert der Bootoption vdso32 von 1 auf 0 geändert, wodurch die 32-Bit vDSO vollständig deaktiviert wird. Dies umgeht zwar den Glibc-Bug, beeinträchtigt aber die Leistung.
Wenn Sie unsicher sind, sagen Sie N: Wenn Sie Ihren eigenen Kernel kompilieren, ist es unwahrscheinlich, dass Sie eine fehlerhafte Version der glibc verwenden.

3.20.32 vsyscall table for legacy applications () →

Legacy-Benutzercode, der nicht weiß, wie er den vDSO finden kann, erwartet, dass er drei Syscalls ausgeben kann, indem er feste Adressen im Kernel-Bereich aufruft. Da dieser Ort nicht mit ASLR randomisiert wird, kann er dazu verwendet werden, die Ausnutzung von Sicherheitslücken zu unterstützen. Diese Einstellung kann zur Boot-Zeit über den Kernel-Befehlszeilenparameter `vsyscall=[emulate|xonly|none]` geändert werden.
Der Emulationsmodus ist veraltet und kann nur noch über die Kernel-Befehlszeile aktiviert werden. Auf einem System mit ausreichend aktueller glibc (2.14 oder neuer) und ohne statische Binärdateien können Sie „None“ ohne Leistungseinbußen verwenden um die Sicherheit zu verbessern.
Wenn Sie unsicher sind, wählen Sie „Nur Ausführung emulieren„.

3.20.32.1 Emulate execution only

CONFIG_LEGACY_VSYSCALL_XONLY [=y] **[Y]**
Der Kernel fängt und emuliert Aufrufe in die feste vsyscall-Adresszuordnung und lässt keine Lesezugriffe zu. Diese Konfiguration wird empfohlen, wenn der Userspace den Legacy-Vsyscall-Bereich verwenden könnte, aber keine Unterstützung für die binäre Instrumentierung von Legacy-Code benötigt wird. Sie entschärft bestimmte Verwendungen des vsyscall-Bereichs als Puffer zur Umgehung von ASLR.

3.20.32.2 None

CONFIG_LEGACY_VSYSCALL_NONE [=n] **[N]**
Es wird überhaupt keine vsyscall-Zuordnung geben. Dies eliminiert jegliches Risiko einer ASLR-Umgehung aufgrund der festen vsyscall-Adressen-Zuordnung. Versuche, die vsyscalls zu verwenden, werden an dmesg gemeldet, so dass entweder alte oder bösartige Userspace-Programme identifiziert werden können.