
TECHNICAL WHITE PAPER

Debian 13 & Proxmox VE auf Huawei TaiShan

Portierung eines produktionsreifen Betriebssystem-Stacks auf die Kunpeng-basierte ARM64-Plattform, inklusive Anpassung der Hardware-Treiber an den Linux-Kernel 7.x.

1 Kurzfassung

Dawico Labs hat Debian 13 „trixie“ auf Huaweis TaiShan-Serverplattform (Kunpeng, ARM64) lauffähig gemacht, sowohl als natives Bare-Metal-System als auch als Virtualisierungs-Host unter Proxmox VE. Der Schwerpunkt lag auf der Anpassung der Hardware-Treiber an den Linux-Kernel 7.x.

Der herstellereigene Treiber **hraid** für den RAID-Controller Huawei SP686C ist vom Hersteller nur für ältere Kernel-Versionen vorgesehen. Unter modernen Kernen ab Version 7.x ließ er sich aufgrund tiefgreifender ABI-Änderungen im Linux-Kernel weder kompilieren noch ausführen. Ohne Anpassung startete das System nicht: Der Controller löste beim Booten einen Kernel-Panic aus.

Durch gezieltes Refactoring der Treiberquellen, die Verankerung im DKMS-System und die Einbindung in angepasste Installationsmedien steht der Controller nun für aktuelle Enterprise-Systeme auf ARM64-Basis zur Verfügung.

Ergebnis: Der Treiber kompiliert und lädt sauber gegen den realen Ziel-Kernel; die RAID-Virtual-Disk ist im Betriebssystem sichtbar und als Root-Laufwerk nutzbar. Debian 13 läuft nativ und unter Proxmox VE auf derselben angepassten Kernel-Basis.

2 Die Plattform: Huawei TaiShan

TaiShan ist Huaweis Serverlinie auf ARM-Basis, angetrieben von der Kunpeng-Prozessorfamilie. Viele Kerne, hohe Speicherbandbreite und eine bemerkenswerte Energieeffizienz machen die Plattform zu einer Grundlage für Cloud-native Workloads, Big Data und verteilte Systeme.

Merkmal	Ausprägung
Architektur	ARM64 (aarch64) mit Huawei-Kunpeng-Prozessoren
Modellvielfalt	1-HE-, 2-HE- und 4-HE-Rack-Server sowie High-Density-Systeme
Prozessoren	1 bis 4 Sockel, bis zu 64 Kerne pro CPU, SMT mit zwei Threads pro Kern bei aktuellen Generationen
Arbeitsspeicher	DDR5 bei aktuellen Generationen, acht und mehr Speicherkanäle pro CPU
Massenspeicher	NVMe-, SAS- und SATA-Konfigurationen mit Hot-Swap
I/O & Netzwerk	PCIe 4.0 und neuer; 10GE/25GE bis 100GE
Verfügbarkeit	Redundante Netzteile und Lüfter (N+1), Hot-Swap-Komponenten

Angaben variieren je nach Modell und Generation (Herstellerangaben). Repräsentative Modelle sind u. a. TaiShan 2280 (Balanced), TaiShan 2480 (High-Performance, 4 Sockel) und TaiShan 5280 (Storage).

3 Zielsetzung: nativ und virtualisiert

Der Betriebssystem-Stack sollte in beiden für die Praxis relevanten Betriebsarten funktionieren, auf identischer, angepasster Kernel-Basis:

- **Natives Debian 13 (Bare-Metal).** Ein schlankes, minimales Debian direkt auf der TaiShan-Hardware, für Server-Workloads, Container und Cloud-native Stacks, ohne Virtualisierungs-Overhead.
- **Proxmox VE.** Der TaiShan-Server als Virtualisierungs-Host für VMs und Container auf ARM64, verwaltet über Proxmox VE, das seinerseits auf einer Debian-Basis aufsetzt.

Beide Wege teilen sich denselben angepassten Kernel-Unterbau. Was für den nativen Betrieb an Treiberarbeit nötig war, kommt damit unmittelbar auch der Virtualisierungsvariante zugute.

4 Die Herausforderung: von Kernel 6 auf Kernel 7

Die eingesetzten Hardware-Treiber waren ursprünglich gegen einen Linux-6-Kernel entwickelt. Beim ersten Kompilierversuch gegen die realen Header des Ziel-Kernels zeigte sich, dass insbesondere die SCSI-Schnittstelle weiter umgebaut wurde als erwartet. Die folgenden Anpassungen wurden empirisch gegen den echten Header-Baum verifiziert, nicht geschätzt.

4.1 Überblick der Kernel-ABI-Änderungen

Bereich	Kernel 6.x	Kernel 7.x (real geprüft)
Unaligned-Header	<asm/unaligned.h>	<linux/unaligned.h>
SCSI-Device-Callbacks	slave_alloc/_destroy/_configure	umbenannt zu sdev_init/_destroy/_configure
Konfigurations-Callback	ein Parameter	zusätzlicher queue_limits-Parameter
Block-Layer-Limits	eigene Setter-Aufrufe	in die queue_limits-API überführt
BSG-Queue-API	5 Parameter	erweitertes Layout mit 6 Parametern

Über Versionsabfragen im Quellcode bleibt der Treiber abwärtskompatibel zu älteren Kernen und lässt sich zugleich nativ gegen Kernel 7.x übersetzen.

4.2 Der RAID-Controller: die größte Hürde

Kern des Problems war der Storage-/RAID-Controller. Fehlten seine SCSI-Device-Callbacks in der Treiber-Vorlage (weil sie im Kernel 7.x anders heißen), wurde beim SCSI-Scan der gerätespezifische Speicher nicht angelegt. Das erste I/O-Kommando dereferenzierte anschließend einen Nullzeiger, mit der Folge eines Kernel-Panic bereits aus dem Initramfs heraus, also noch vor dem eigentlichen Systemstart:

```
/* SCSI-Scan ohne aktive Device-Callbacks: hostdata bleibt NULL */
hostdata = sdev->hostdata;
io_cmd.rw.hdid = cpu_to_le32(hostdata->hdid); // NULL-Deref -> KERNEL PANIC
```

Die Lösung: Die betroffenen Callbacks bleiben vollständig aktiv, lediglich unter ihren neuen Namen für Kernel 7.x, und die veralteten Block-Layer-Aufrufe innerhalb der Gerätekonfiguration werden auf die neue `queue_limits`-API umgestellt. Damit wird der gerätespezifische Speicher wieder korrekt allokiert und der Panic entfällt.

4.3 Korrekte CPU-Erkennung

Eine weitere Baustelle war die zuverlässige Erkennung und Initialisierung der Kunpeng-Prozessoren auf der ARM64-Plattform. Erst wenn die CPU-Topologie sauber erkannt wird, arbeiten Scheduler, Interrupt-Verteilung und die hardwarenahen Treiber stabil zusammen, die Grundlage für den produktiven Betrieb. Dazu gehört SMT (Simultaneous Multithreading, das Hyper-Threading-Prinzip): Aktuelle Kunpeng-Prozessoren führen zwei Threads pro Kern aus, und alle logischen Prozessoren stehen dem Betriebssystem vollständig zur Verfügung.

5 Betrieb und Wartung

Ein einmalig übersetzter Treiber genügt im Produktivbetrieb nicht. Zwei Bausteine sorgen dafür, dass die Lösung dauerhaft und ohne Handarbeit trägt.

5.1 Update-Sicherheit durch DKMS

Der Treiber wurde in ein DKMS-Paket überführt. System- und Kernel-Updates bauen das Modul im Hintergrund automatisch für den jeweils aktiven Kernel neu. Damit entfällt der manuelle Wartungsaufwand, der solche Speziallösungen sonst schnell zum Risiko macht: Nach einem Kernel-Update bootet das System weiterhin zuverlässig von seinem RAID-Volume.

5.2 Boot- und Installer-Integration

Damit das Betriebssystem direkt auf dem RAID-Volume installiert werden kann, wurde das kompilierte Modul inklusive Modul-Index in die Live-Umgebungen und Ramdisks der Installationsmedien eingebunden. Das RAID-Array steht bereits beim Booten des Installers als Laufwerk zur Verfügung und kann unmittelbar als Root-Laufwerk verwendet werden, ohne Zwischenschritte oder nachträgliches Einspielen von Treibern.

6 Verifikation auf echter Hardware

Die Anpassungen wurden nicht simuliert, sondern gegen den echten Kernel auf dem Zielsystem kompiliert und geladen. Das Kernel-Modul meldet die korrekte Kernel-Signatur für die aarch64-Zielumgebung:

```
$ modinfo hiraidd.ko | grep vermagic
vermagic: 7.x SMP preempt mod_unload modversions aarch64
```

Nach dem Laden des Moduls erkennt der Kernel den Controller, und die RAID1-Virtual-Disk wird als Blockgerät eingebunden:

```
$ insmod hiraidd.ko
scsi host8: hiraidd
scsi 8:2:0:0: Direct-Access      HUAWEI  RAID1          C00
sd 8:2:0:0: [sdc] 1874335744 512-byte logical blocks: (960 GB/894 GiB)
sd 8:2:0:0: [sdc] Attached SCSI disk
```

Nachweis: Die RAID1-Virtual-Disk des Controllers ist sichtbar und ansprechbar. Das System bootet stabil vom RAID-Volume, der zuvor auftretende Kernel-Panic tritt nicht mehr auf.

7 System-Kompatibilität und Status

Da die Anpassungen an den generischen Kernel-ABIs ansetzen, lässt sich die Lösung plattformübergreifend auf ARM64-Architekturen einsetzen:

Plattform	Status	Umsetzung
Proxmox VE 9.2 (ARM64)	Erfolgreich verifiziert	DKMS-Integration und angepasster ISO-Installer
Debian 13 (ARM64)	Erfolgreich verifiziert	DKMS-Paket und angepasste Netinst-Medien; Installation und Boot vom RAID-Volume auf echter Hardware bestätigt
Ubuntu 26.04 LTS	Erfolgreich verifiziert	Angepasster Installer mit eingebundenem hiraidd-Treiber

Die Installations-Images für Debian 13, Proxmox VE und Ubuntu 26.04 LTS stehen unter **huawei.dawico.dev** zum Download bereit. Den Stand der Portierungen zeigt die Roadmap auf derselben Seite.

8 Über Dawico Labs

Dawico Labs ist die Forschungs- und Entwicklungseinheit von Dawico Systems. Schwerpunkte sind hochskalierbare HA-Systeme, Web- und Cloud-Engineering, globale App-Projekte und KI-Systeme mit GPU-Power. Was sich im Labor bewährt, geht als Blaupause in die Produktion.

Kontakt: huawei.dawico.dev · dawico.dev