

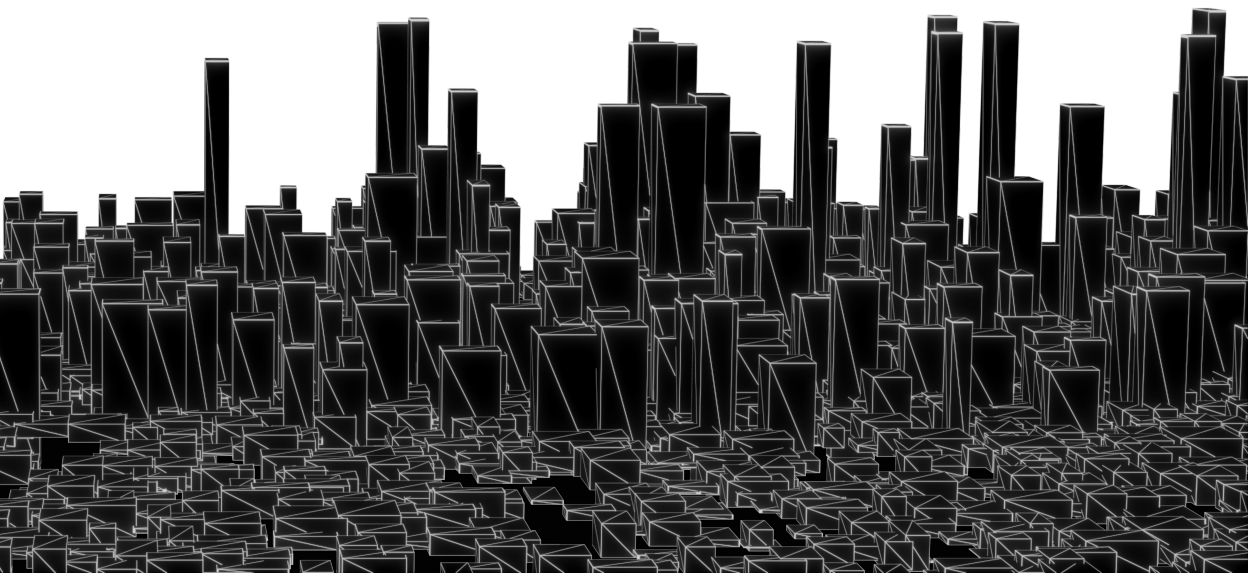
Hagen Paul Pfeifer

Linux-Systemanalyse

Von High-Level-Architekturanalysen zu Low-Level-Code-Optimierungen

1. Auflage

München – 2025



Ich hoffe, dass Sie Freude an diesem Buch haben und Ihre Erwartungen erfüllt werden. Ihre Anregungen und Kommentare sind jederzeit willkommen. Bitte nutzen Sie die Kontaktinformationen auf der Buch-Homepage: <https://jauu.net/linux-analyse/>.

Bibliografische Informationen der Deutschen Nationalbibliothek: Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <https://dnb.dnb.de> abrufbar.

Das vorliegende Buch ist in all seinen Teilen urheberrechtlich geschützt. Alle Rechte vorbehalten. Die Verwendung der Texte und Illustrationen, auch auszugsweise, ist ohne schriftliche Zustimmung des Autors urheberrechtswidrig und daher strafbar. Die automatische Analyse des Werkes, um daraus Informationen insbesondere über Muster, Trends und Korrelationen gemäß §44b UrhG („Text und Data Mining“) zu gewinnen, ist untersagt.

Ungeachtet der Sorgfalt, die auf die Erstellung von Text, Abbildungen und Programmen verwendet wurde, kann der Autor für mögliche Fehler und deren Folgen keine juristische oder sonstige Haftung übernehmen.

Die in diesem Werk wiedergegebenen Gebrauchsnamen, Handelsnamen und Warenbezeichnungen usw. können auch ohne besondere Kennzeichnung Marken sein und als solche den gesetzlichen Bestimmungen unterliegen.

Copyright © 2022 – 2025 Hagen Paul Pfeifer

1. Auflage 2025
Version: 2025-09-28

Verlag: BoD - Books on Demand GmbH, In de Tarpen 42, 22848 Norderstedt, bod@bod.de
Lektorat: Roberta Martin, Berlin
Druck: Libri Plureos GmbH, Friedensallee 273, 22763 Hamburg

ISBN: 978-3-8192-1292-5

Inhaltsverzeichnis

Vorwort	xv
1. Einführung	1
1.1. Verwendetes Setup	2
1.1.1. Software	2
1.1.2. Hardware	2
1.2. Abkürzungen und Begrifflichkeiten	5
1.3. Build Identifikation	9
1.4. Analyse und Optimierungen	13
1.4.1. Leistungsüberwachung und Optimierung	14
1.4.2. Fehlerbehebung	14
1.4.3. Ressourcenprofiling und Kapazitätsplanung	14
1.4.4. Sicherheitsanalyse und Auditing	15
1.4.5. Applikations- und Systemverständnis	15
I. Grundlagen und Einführung Werkzeuge	17
2. Perf	19
2.1. Hintergrund und Historie	19
2.2. Voraussetzungen	21
2.3. Hello World	22
2.3.1. Berechtigungen	23
2.4. Perf Events	27
2.4.1. Hardware Events	28
2.4.2. Software Events	38
2.4.3. Tracepoint Events	39
2.4.4. Dynamische Tracepoints	43
2.4.5. Statische Probepoints	45
2.4.6. Metriken und Metrik-Gruppen	47
2.4.7. Hardware Breakpoint	51

Inhaltsverzeichnis

2.4.8.	Strukturierte Event-Ausgabe im JSON-Format	53
2.4.9.	Zusammenfassung	54
2.5.	Record und Report	54
2.5.1.	Beispielobjekt Branching Spinner	55
2.5.2.	Perf Record	58
2.5.3.	Events Spezifizieren	61
2.5.4.	Heterogene Kernarchitekturen auf ARM- und x86-Systemen	64
2.5.5.	Event Modifier	65
2.5.6.	Aufnahmemodi und Bereichsbegrenzung	65
2.5.7.	Workloads Definieren	71
2.5.8.	Event Gruppen und Leader	72
2.5.9.	Filter	76
2.5.10.	Weitere Aufnahme-Konfigurationsmöglichkeiten	84
2.5.11.	Aufnahmen Analysieren – Reports	85
2.5.12.	Stichprobenentnahme – Sampling	88
2.5.13.	Latenzprofiling	96
2.5.14.	Call Graphs	100
2.5.15.	Call Graph Verstehen	116
2.5.16.	Anpassen der Ausgabe	123
2.5.17.	Zeitbezogene Analysefunktionen	138
2.5.18.	Fokussierte Aufzeichnung	143
2.5.19.	CPU-Affinität	147
2.6.	Annotate	149
2.7.	Script	151
2.7.1.	Einführung	151
2.7.2.	Event Pager	152
2.7.3.	Scripting Framework	156
2.7.4.	Stolpersteine	160
2.8.	Dynamische Tracepoints – Perf Probe	163
2.8.1.	Userspace Probing	164
2.8.2.	Kernelspace Probing	179
2.8.3.	User Statically Defined Tracepoints	181
2.8.4.	User Events	187
2.9.	Task Scheduling Eigenschaften – Sched	191
2.9.1.	Record	194
2.9.2.	Latency	196
2.9.3.	Timehist	199
2.9.4.	Map	203
2.9.5.	Replay	206

2.9.6. Script	207
2.10. Stat	207
2.10.1. Event-Selektor	210
2.10.2. Filterung	211
2.10.3. Nützliche Optionen	212
2.10.4. Pre- und Post-Kommandierung	215
2.10.5. Dezimaltrennzeichen und Tausendergruppierung	216
2.11. Trace	218
2.11.1. Einordnung und Architektur	218
2.11.2. Tracing von Systemcalls	219
2.11.3. Tracing von Kernel-Events und Tracepoints	220
2.11.4. Aggregations- und Gruppierungsoptionen	221
2.11.5. Zusammenfassung und Performancevergleich	223
2.12. Ftrace	223
2.13. Kwork	225
2.13.1. Daten-Aufzeichnung	225
2.13.2. Auswertung	226
2.14. Bench	230
2.15. Mem	234
2.16. Lock	236
2.16.1. Kernel-space Lock-Analyse	237
2.16.2. Userspace Lock-Analyse	242
2.17. Inject	249
2.17.1. Aktualisierung von Build-ID-Informationen	250
2.17.2. Task-Schlafzeiten Analysieren	250
2.17.3. Abschließende Bemerkungen	253
2.18. Diff	254
2.18.1. Vergleichsmethoden	254
2.18.2. Beispiel	255
2.19. Data	258
2.19.1. Common-Trace-Format	258
2.19.2. JSON	259
2.20. Ergänzende Werkzeuge	260
2.20.1. Archive	261
2.20.2. Config	263
2.20.3. API Interface	264
2.21. Frontends	268
2.21.1. Hotspot	269
2.21.2. Traceshark	273

Inhaltsverzeichnis

2.21.3. Kernelshark	276
2.21.4. LTTng und Trace Compass	278
2.21.5. Perfetto	279
2.22. Zusammenfassung	281
3. eBPF	283
3.1. Ursprung und Evolution von eBPF	284
3.2. Grundlagen	288
3.3. Hello World	290
3.4. Struktur und Komponenten von eBPF	294
3.4.1. Programtypen	294
3.4.2. Hilfsfunktionen für eBPF-Programme	295
3.4.3. Maps - eBPF Datenkontainer	298
3.5. eBPF CO-RE	303
3.6. Entwicklungsumgebungen	304
3.6.1. Rust	305
3.6.2. Go	306
3.6.3. Python	306
3.7. Standardisierung	308
3.8. BPF Type Format - BTF	309
3.9. Tools	312
3.9.1. BCC	312
3.9.2. Bpftrace	313
3.9.3. PWRU	313
3.9.4. Tracee	315
3.9.5. Retsnoop	316
3.9.6. Kmemsnoop	316
3.9.7. Parca	318
3.9.8. Below	318
3.10. Zusammenfassung	320
4. System-Tracer	321
4.1. Intel Processor Trace	322
4.1.1. Perf	324
4.1.2. GDB	341
4.1.3. Simple PT	346
4.1.4. IPT Decoder Bibliothek	347
4.2. ARM CoreSight	348
4.2.1. Embedded Trace Macrocell	349

4.2.2.	Voraussetzungen	349
4.2.3.	Aufzeichnung und Verwendung	350
4.2.4.	Zusätzliche Informationen	351
4.3.	Zusammenfassung	352
II.	Analyse-Subsysteme und Vertiefung	353
5.	CPU	355
5.1.	Einführung in die Moderne CPU-Architektur	356
5.1.1.	Grundlagen der CPU	356
5.1.2.	Prozessor Core	359
5.1.3.	Prozessor Frontend	361
5.1.4.	CPU-Backend und Out-of-Order-Ausführung	369
5.1.5.	Speicherkonsistenzmodelle	378
5.1.6.	Zusätzliche Hinweise	380
5.2.	Analyse	383
5.2.1.	Messung von Branch Mispredictions	383
5.2.2.	Frontend Bubbles – Übermäßige MSROM-Nutzung	390
5.2.3.	Cache- und Speicherinterferenzen	393
5.3.	Zusammenfassung	396
6.	Task Scheduling	399
6.1.	Grundlagen	400
6.1.1.	Policies – Scheduler-Algorithmen	400
6.1.2.	Prioritäten und Nice Level	406
6.1.3.	Ausführungskontexte	408
6.1.4.	Deferred Work	411
6.1.5.	Unterbrechbarkeit und Schedulingzeitpunkte	417
6.1.6.	Modifikation des Task-Scheduling	421
6.2.	Analyse	424
6.2.1.	Scheduling-Standardtools	424
6.2.2.	Scheduler Tracepoints	429
6.2.3.	Latenz- und Realtime-Analyse	441
7.	Speicher	453
7.1.	Grundlagen	453
7.1.1.	Virtuelle- und physikalische Adressräume	454
7.1.2.	Buddy-System und Per-CPU Allokatoren	457

Inhaltsverzeichnis

7.1.3.	Huge Pages	461
7.1.4.	Slab Allokator	463
7.1.5.	Page Cache	466
7.1.6.	Weiterführende Informationen	469
7.1.7.	Allokatoren-Analyse	471
7.2.	Prozessspeicher	487
7.2.1.	Allokationsanpassungen	488
7.2.2.	Speicherverbrauchsanalyse von Prozessen	489
7.2.3.	Speicherdiagnosetools	490
7.2.4.	Stack-Allokationen	497
7.2.5.	Heap-Allokationen	501
7.2.6.	Allokationen verfolgen	502
7.3.	Hardware-Speicherzugriffe Monitoring	508
7.3.1.	Perf	509
7.3.2.	Damon Framework	528
7.3.3.	Hilfswerkzeuge zur Speicheranalyse	533
7.4.	Abschließendes Beispiel	536
7.4.1.	Problembeschreibung und Hypothese	536
7.5.	Zusammenfassung	542
8.	Dateisystem	545
8.1.	Grundlagen	546
8.1.1.	Applikationen	547
8.1.2.	Virtual File System	548
8.1.3.	Dateisystem	548
8.1.4.	Block IO – BIO	549
8.1.5.	Block Layer	550
8.1.6.	Device-Treiber	552
8.1.7.	Physikalische Geräte	553
8.2.	Dateizugriffe und Datenstrukturen	553
8.2.1.	In-Memory-Strukturen	554
8.2.2.	On-Disk-Strukturen	557
8.2.3.	Page Cache	562
8.2.4.	Dentry und Inode Cache	564
8.2.5.	Direct IO	565
8.3.	Filesystem-Analyse	567
8.3.1.	Maßgeschneiderte Lösungen - procfs und sysfs	567
8.3.2.	IOTop	573
8.3.3.	IOStat	574

8.3.4.	Tracepoints	574
8.3.5.	BCC Tools	579
8.3.6.	IO Pressure Stall Information	583
8.3.7.	Blktrace	584
8.3.8.	Zusammenfassung	592
9.	Zeit und Timer	595
9.1.	Einführung	596
9.2.	High Resolution Timer	597
9.3.	Clock Sources und Clock Events	598
9.4.	Tickless	603
9.5.	Timer-Lebenszyklus	607
9.6.	Timer-Analyse	608
9.7.	Task Timer Slack	614
9.8.	Manueller Timer Slack	615
9.9.	Abschließendes	617
9.9.1.	Überblick Zeit und Zählregister	617
9.9.2.	Virtual Dynamic Shared Object	618
9.9.3.	Alarm Timer	620
10.	Energiemanagement	623
10.1.	Einführung	623
10.1.1.	Elektrotechnische Grundlagen	625
10.1.2.	Energetische Grundsätze	627
10.1.3.	Konfigurationsstandard ACPI	628
10.1.4.	Hardware-Design-Überlegungen	630
10.2.	x86-64	631
10.2.1.	System-Ruhezustände – S States	632
10.2.2.	Prozessor Ruhezuständen – C-States	637
10.2.3.	CPU-Leistungsskalierung und P States	652
10.2.4.	Uncore und Peripherie	666
10.2.5.	Echtzeitanforderungen und Power Management	683
10.2.6.	Opportunistic Sleep	684
10.3.	ARM	689
10.3.1.	Device Trees	690
10.3.2.	Uncore und Peripherie	691
10.4.	Weiterführende Werkzeuge	694
10.4.1.	Powerletrics	694
10.4.2.	Green Metrics Tool	695

Inhaltsverzeichnis

10.4.3. i7z	697
10.4.4. Powertop	697
10.4.5. Turbostat	698
10.4.6. TLP	699
10.5. Energiebewusstes Applikationsdesign	699
11. Methodologie	703
11.1. Anforderungen und Ziele definieren	704
11.2. Anforderungen und Vergleichbarkeit	705
11.3. Kontinuierliche System-Kartierung und Baselineing	709
11.4. Datenhaltung	710
11.4.1. Speicherung und Nachvollziehbarkeit	710
11.4.2. Strukturierung und Nachverarbeitung	712
11.5. Analyse-Werkzeugkasten	713
11.6. Analysemethoden und Vorgehensweisen	714
11.7. Anlassgetriebene vs. kontinuierliche Analyse	715
11.7.1. Kontinuierliche Analyse – Proaktive Analyse	715
11.7.2. Anlassgetriebene Analyse – Reaktive Analyse	716
11.7.3. Gegensatz und Synergie	716
11.8. Kommunikation von Analyseergebnissen	717
11.9. Integration in Entwicklungszyklen	718
11.9.1. Schnittstellen	719
11.9.2. Collect-, Preprocess- und Analyse-Zyklus	719
11.10 Zusammenfassung	723
III. Anhang	725
A. Beispielverzeichnis	727
B. Benchmarking und Lastgeneratoren	729
B.1. Ziele und Aufgaben von Benchmark-Tools	730
B.2. Kategorien von Benchmarks	730
B.3. Professionelles Benchmarking	731
B.4. Benchmark-Tools für verschiedene Einsatzbereiche	732
B.5. CPU	732
B.5.1. Perf Bench	732
B.5.2. Sysbench	732
B.5.3. 7-Zip	733

B.5.4. Stress-ng	734
B.6. GPU	737
B.6.1. GLmark2	737
B.6.2. Blender Benchmark	738
B.7. Netzwerk	738
B.7.1. iperf	738
B.7.2. netperf	739
B.7.3. pktgen	740
B.8. RAM	742
B.8.1. Stream	742
B.8.2. Ramspeed	744
B.8.3. likwid-bench	745
B.8.4. Sysbench Memory Test	747
B.9. I/O	748
B.9.1. dd	748
B.9.2. Flexible I/O Tester - FIO	748
B.10. Verschiedenes	750
B.10.1. Phoronix Test Suite	750
B.10.2. Geekbench	751
C. Individuelle Installationen	755
C.1. Debian Virtuelle Maschine	755
C.1.1. Debian Bootstrap	757
C.1.2. Debian VM starten	761
C.1.3. Debian VM anpassen	762
C.2. Individueller Linux-Kernel	763
C.3. Linux-Kernel interaktiv Debuggen	770
C.4. Spezifisches Perf	774
C.5. Intel XED	776
C.6. Real-Time Test Suite und Hackbench	777
D. Linux-Entwicklung verfolgen	781
D.1. Linux Kernel Mailing List	781
D.1.1. E-Mail	782
D.1.2. Lore, Lei und B4	784
D.2. Linux-Kernelquellen	787
D.2.1. Cscope	787
D.2.2. Elixir	788
D.2.3. Git Log	789

D.3. Blogs and weitere Medien	790
E. Binärcode-Analyse	791
E.1. Readelf	791
E.2. Objdump	800
E.3. Radare2	803
E.4. Abschließende Bemerkungen	812
Literaturverzeichnis	814
Stichwortverzeichnis	831