ISE SRA

Leibniz Universität Hannover

Analyzing and Optimizing TLB-Induced Thread Migration Costs on Linux/ARM

Tobias Landsberg

21. November 2019

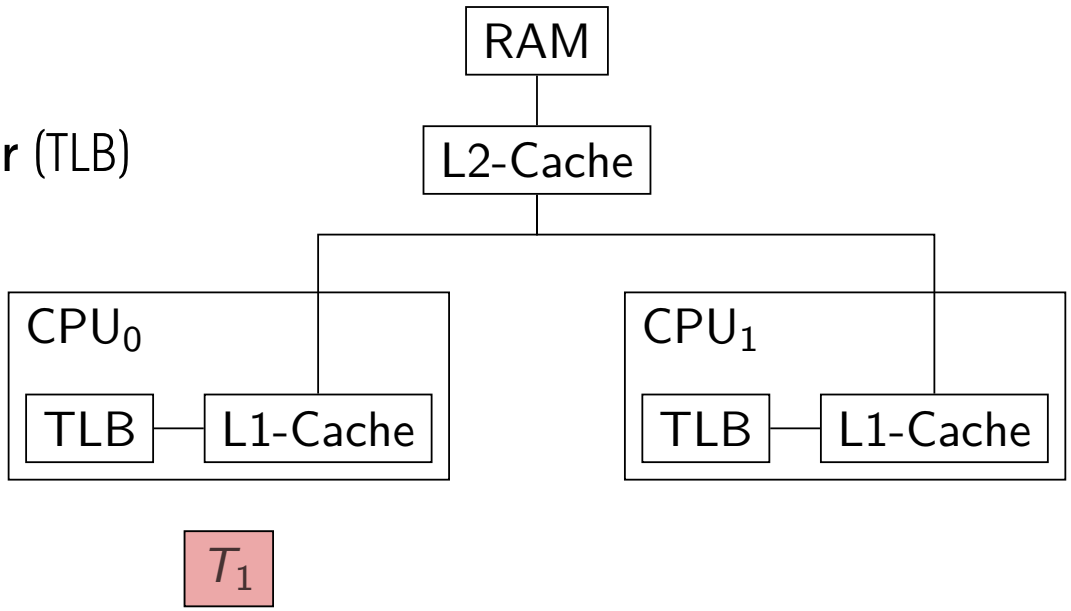
Motivation

- Speicherzugriffe sind langsam
  - $\approx$ 150 Takte[1]
- Bottleneck moderner CPUs
- Caches als Lösung
  - Hierarchien (L1, L2 etc.)
    - 3 Takte für L1-Zugriff[1]
  - **Translation Lookaside Buffer** (TLB)

[1] Gemessen auf einem Raspberry Pi 2

Motivation

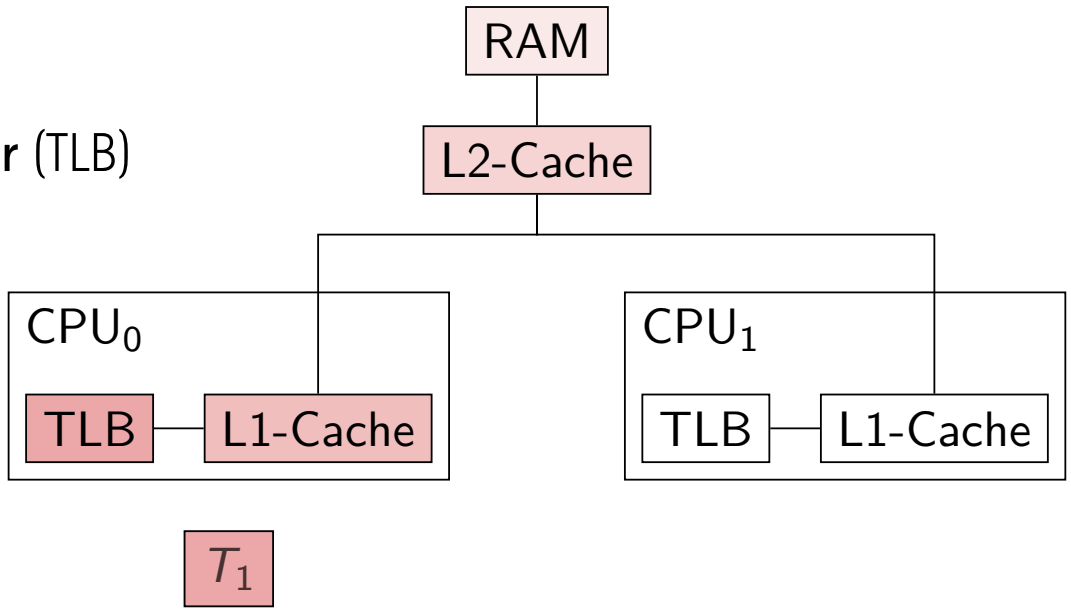
- Speicherzugriffe sind langsam
  - $\approx$ 150 Takte[1]
- Bottleneck moderner CPUs
- Caches als Lösung
  - Hierarchien (L1, L2 etc.)
    - 3 Takte für L1-Zugriff[1]
  - **Translation Lookaside Buffer** (TLB)



[1] Gemessen auf einem Raspberry Pi 2

Motivation

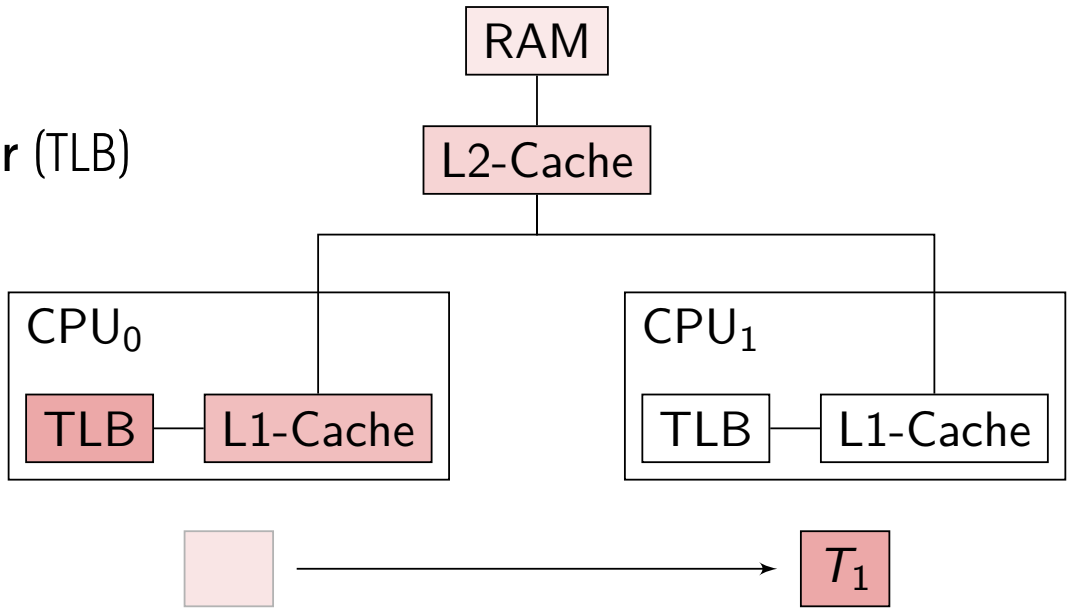
- Speicherzugriffe sind langsam
  - $\approx$ 150 Takte[1]
- Bottleneck moderner CPUs
- Caches als Lösung
  - Hierarchien (L1, L2 etc.)
    - 3 Takte für L1-Zugriff[1]
  - **Translation Lookaside Buffer** (TLB)



[1] Gemessen auf einem Raspberry Pi 2

Motivation

- Speicherzugriffe sind langsam
  - $\approx$ 150 Takte[1]
- Bottleneck moderner CPUs
- Caches als Lösung
  - Hierarchien (L1, L2 etc.)
    - 3 Takte für L1-Zugriff[1]
  - **Translation Lookaside Buffer** (TLB)
- Verlust von Cacheinhalten bei Wechsel der CPU



[1] Gemessen auf einem Raspberry Pi 2

Motivation (2)

- **Lösung**: Migration des TLBs zusammen mit Thread
- TLB speichert Übersetzungen zwischen
- Zugriff auf virtuelle Adresse braucht insgesamt drei Speicherzugriffe

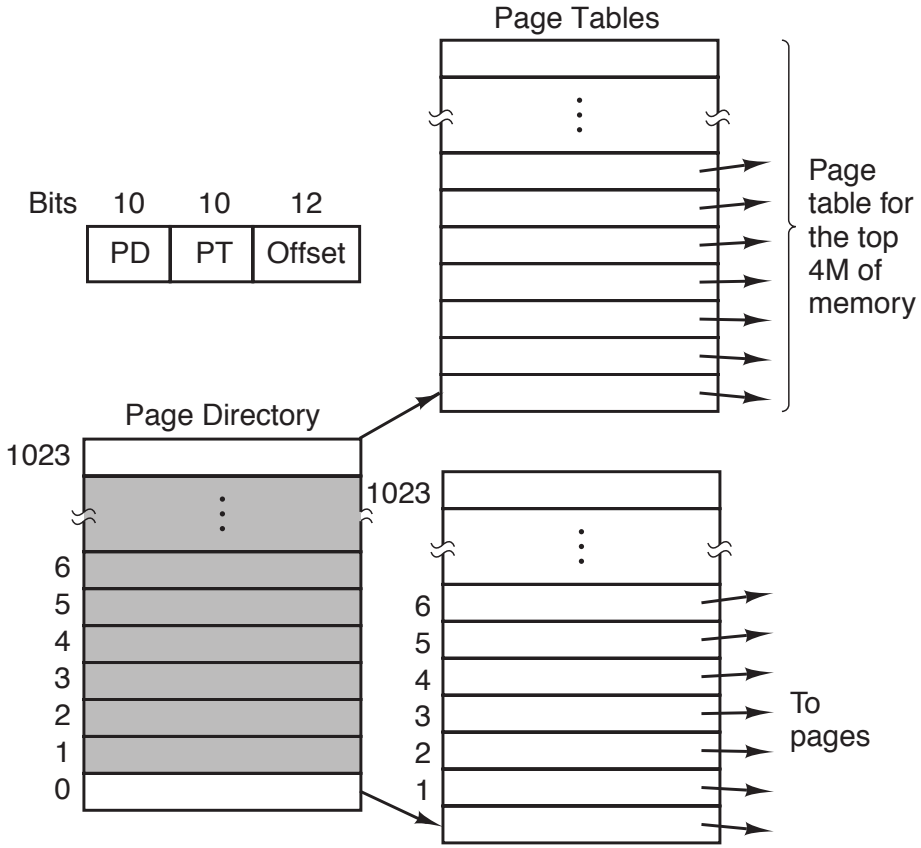


Abbildung: Adressübersetzung (nach Tanenbaum)

Inhalt

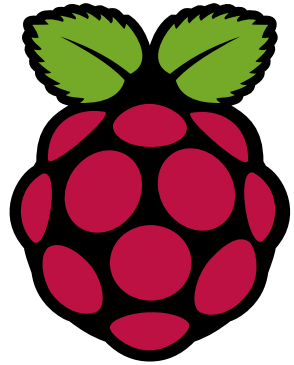
TLB-Migration

Hardwareanpassungen

Linux-Integration

Fazit

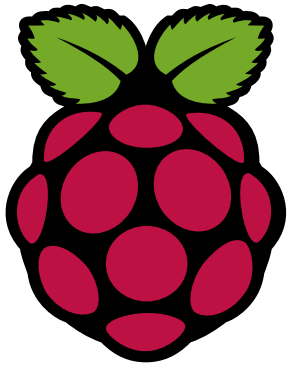
Technische Umsetzung



Ziel: Migration des TLBs zusammen mit restlichem Kontext

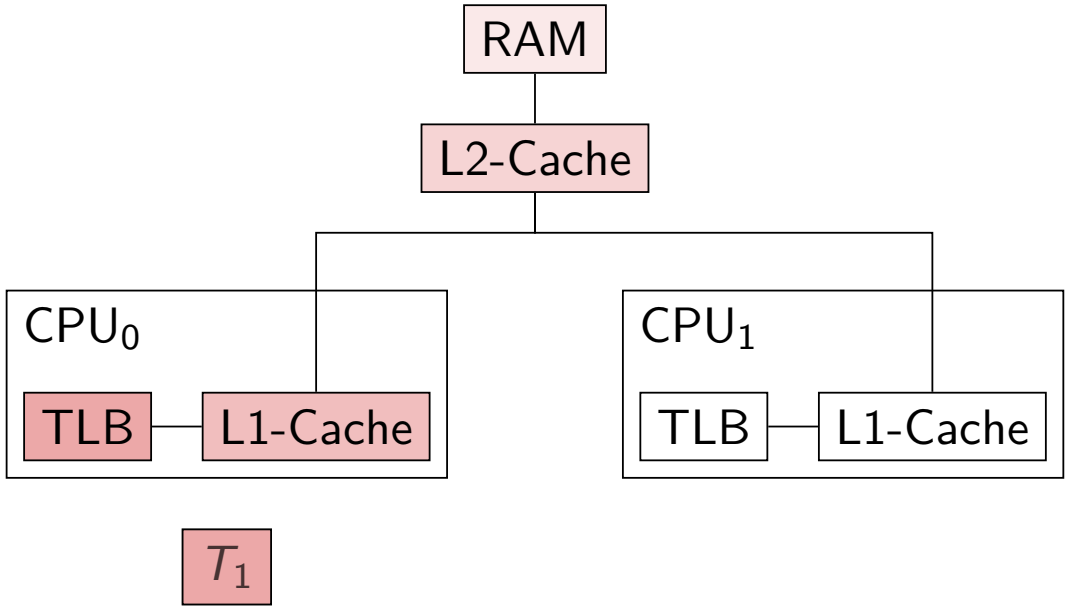
- Plattform: Raspberry Pi 2 mit ARM Cortex-A7 Quadcore-CPU
- Bei Thread-Migration aktuellen TLB-Zustand sichern und später wiederherstellen

Technische Umsetzung

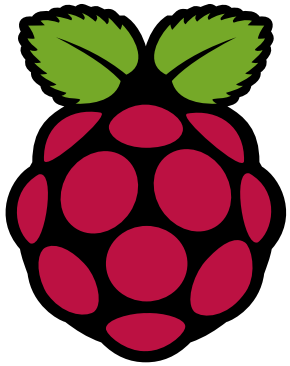


Ziel: Migration des TLBs zusammen mit restlichem Kontext

- Plattform: Raspberry Pi 2 mit ARM Cortex-A7 Quadcore-CPU
- Bei Thread-Migration aktuellen TLB-Zustand sichern und später wiederherstellen

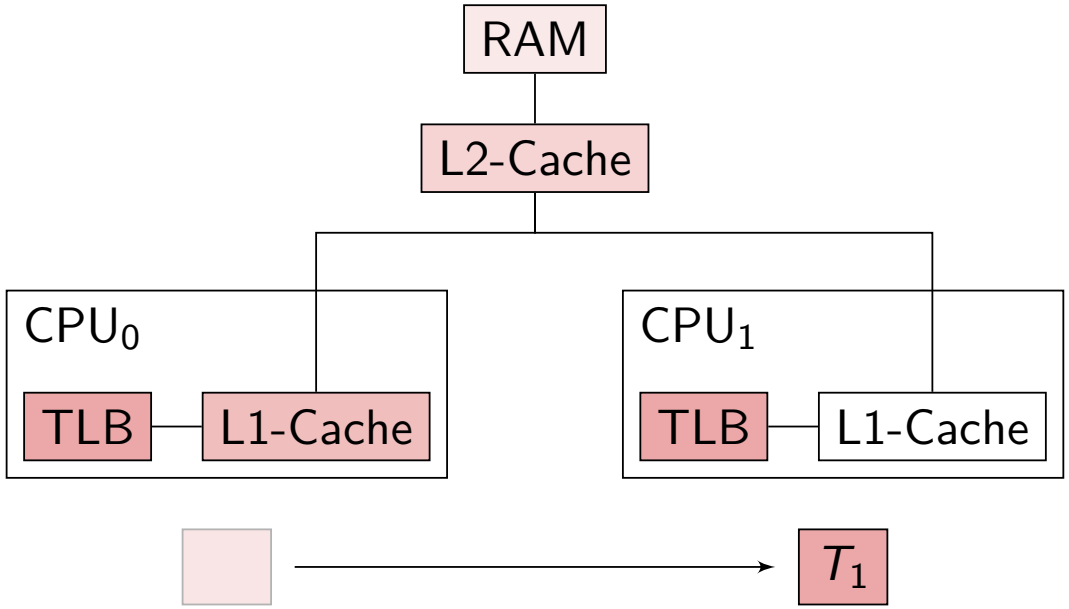


Technische Umsetzung

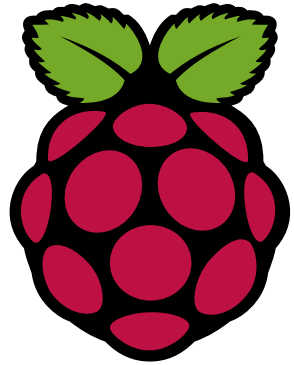


Ziel: Migration des TLBs zusammen mit restlichem Kontext

- Plattform: Raspberry Pi 2 mit ARM Cortex-A7 Quadcore-CPU
- Bei Thread-Migration aktuellen TLB-Zustand sichern und später wiederherstellen



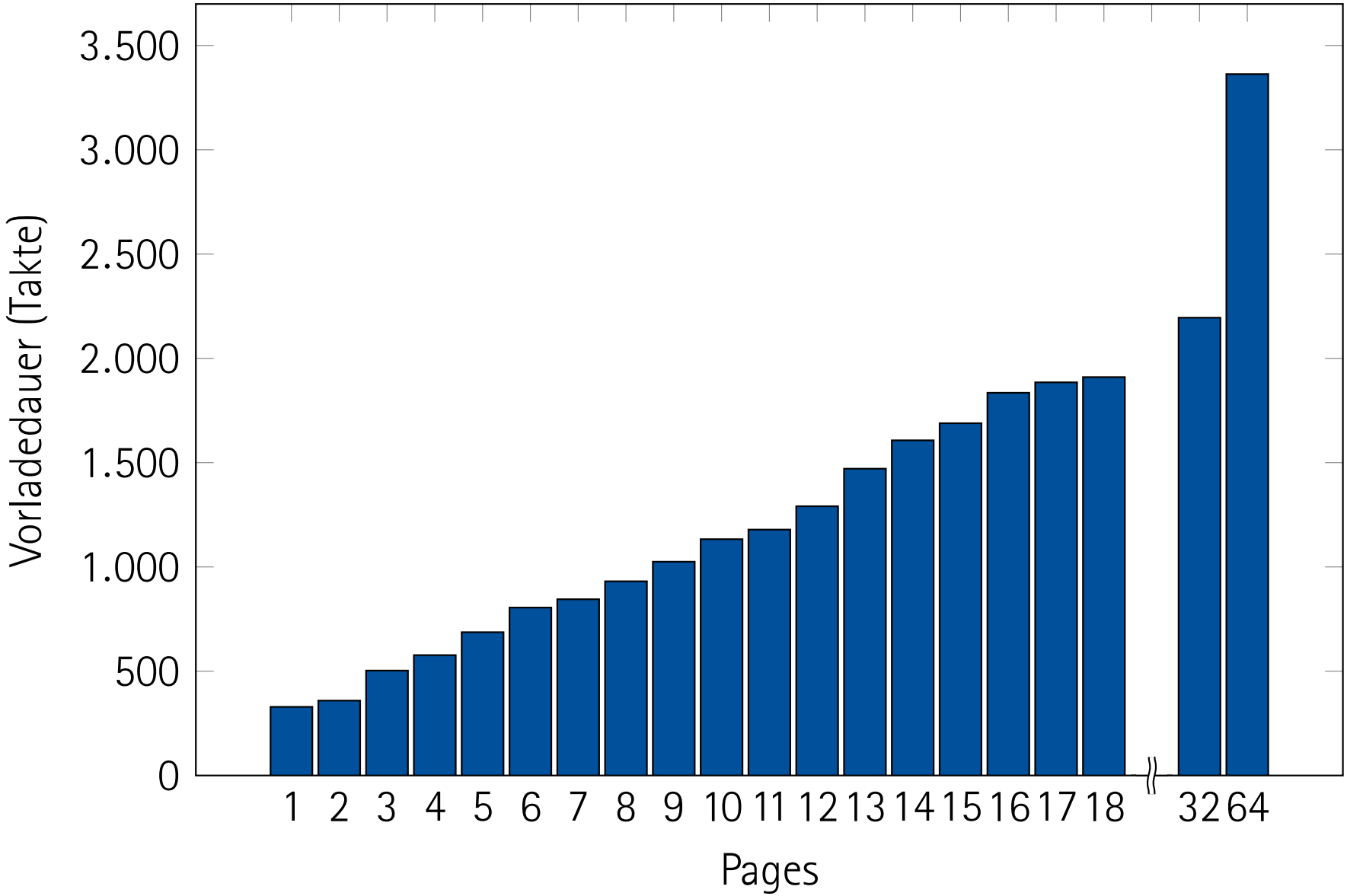
Technische Umsetzung



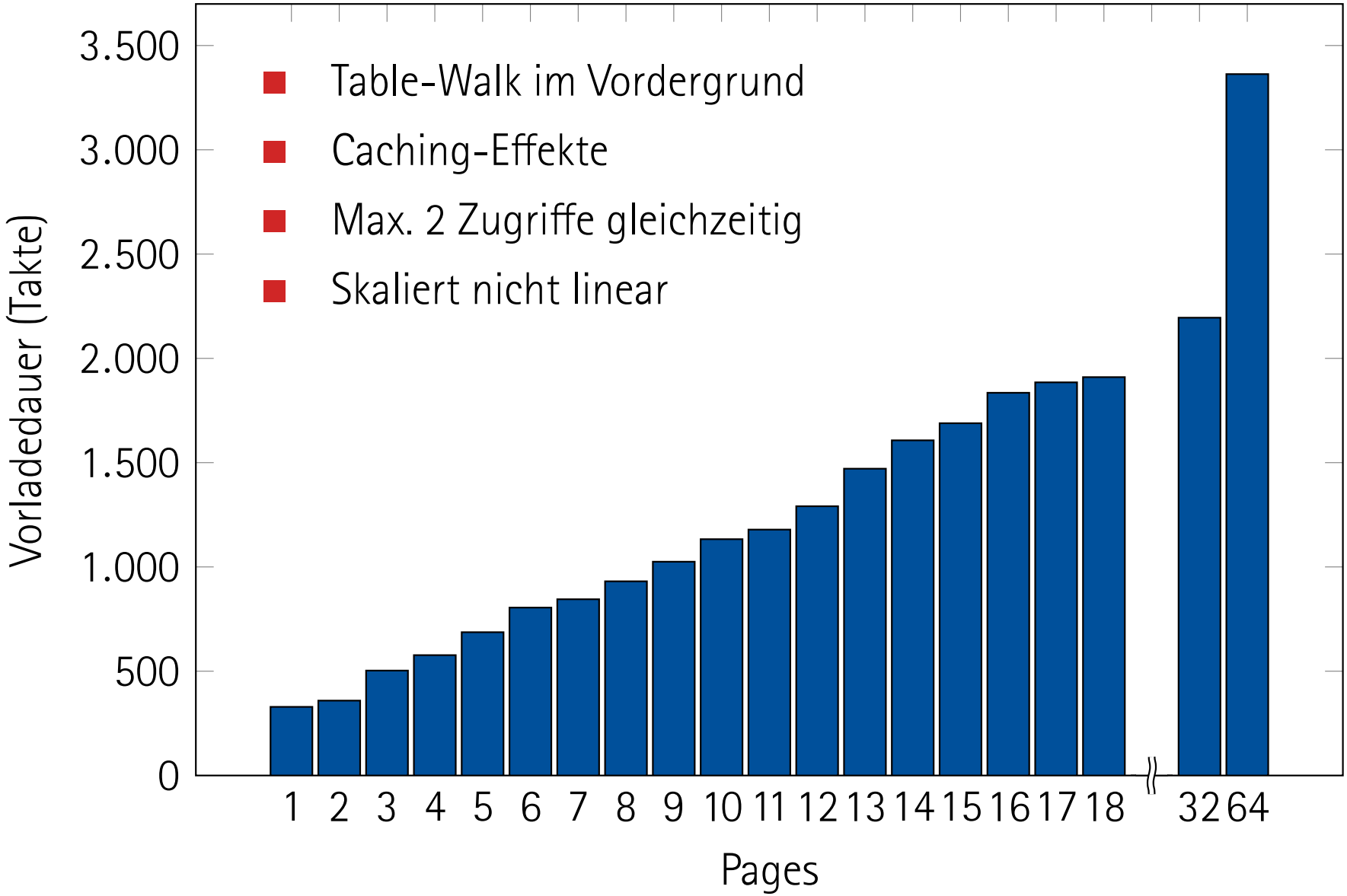
Ziel: Migration des TLBs zusammen mit restlichem Kontext

- Plattform: Raspberry Pi 2 mit ARM Cortex-A7 Quadcore-CPU
- Bei Thread-Migration aktuellen TLB-Zustand sichern und später wiederherstellen
- TLB-Zugriff
  - Lesend: `mcr p15, 3, <Rt>, c15, c4, 2`
  - Schreibend (indirekt): `pld [<Rn>]`

Verhalten der pld-Instruktion



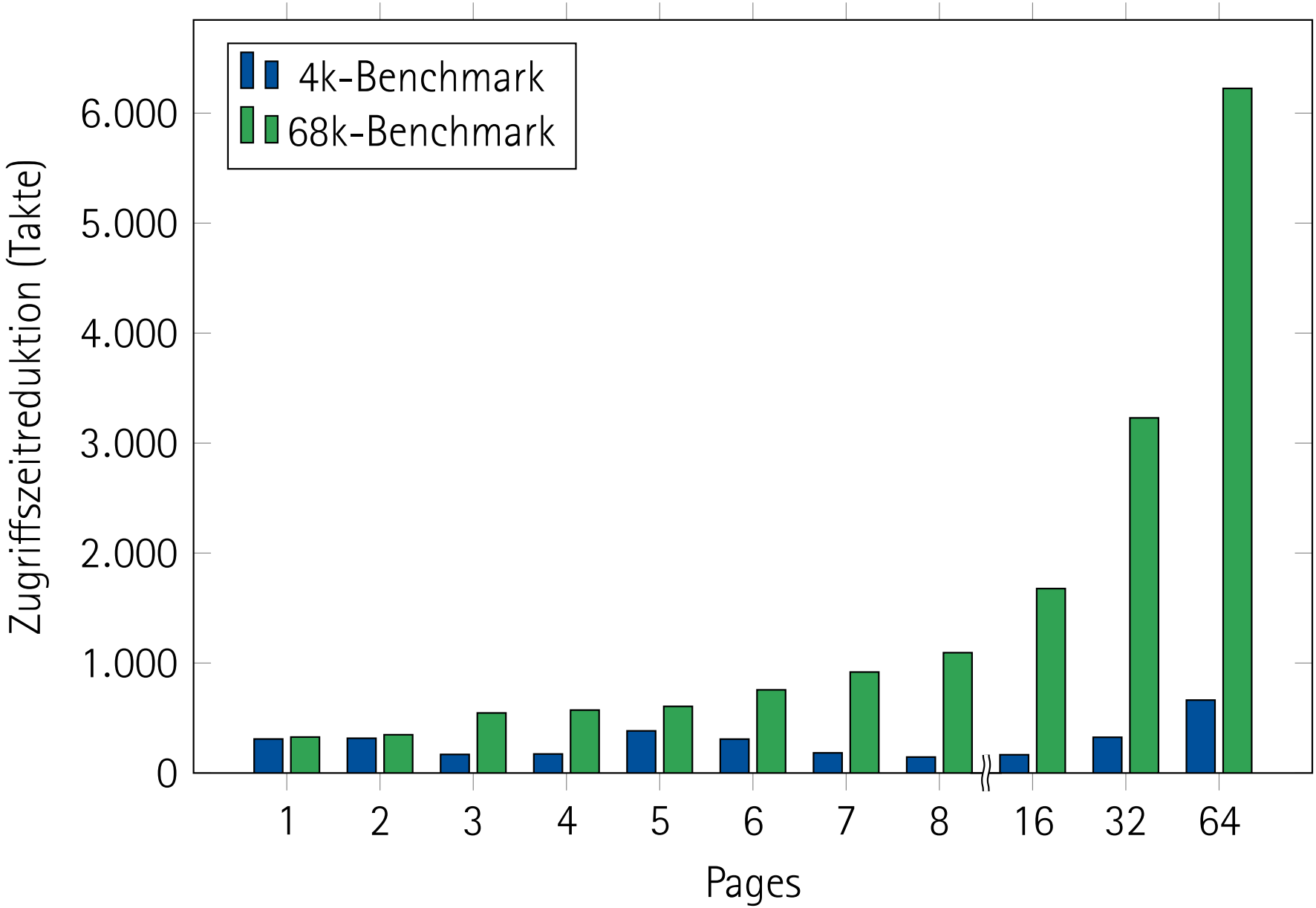
Verhalten der pld-Instruktion



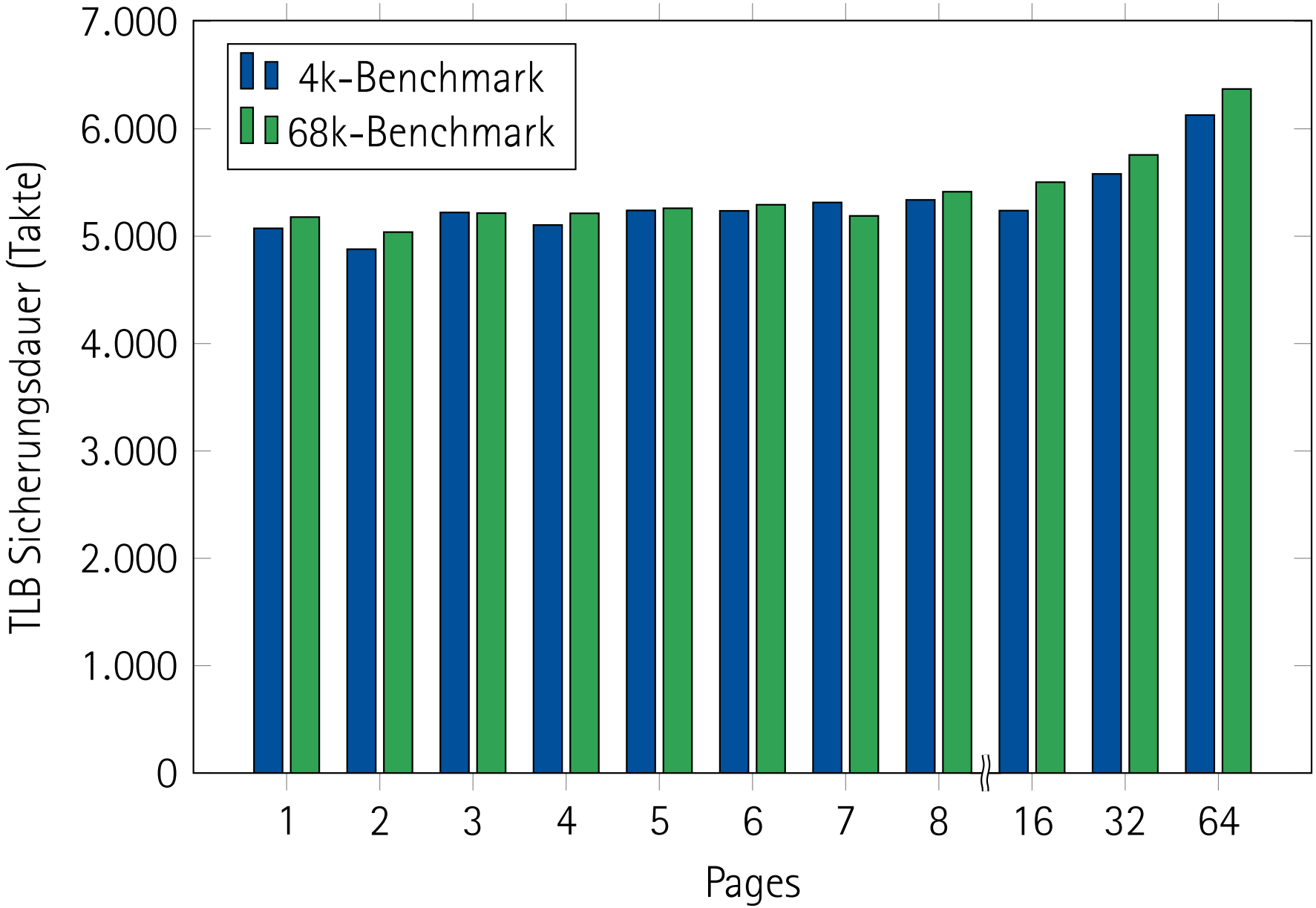
Benchmarks

- Zwei Benchmarks als Kernel-Modul für Linux
  1. Datencaches und TLB leeren
  2. Auf *N* Pages im Abstand von 4 bzw. 68 KiB lesend zugreifen
  3. Relevante TLB-Einträge im Speicher sichern
  4. Datencaches und TLB leeren
  5. TLB mittels `pld`-Instruktion wiederherstellen
  6. Auf *N* Pages im selben Abstand lesend zugreifen
- Bezeichnet als 4k- und 68k-Benchmark

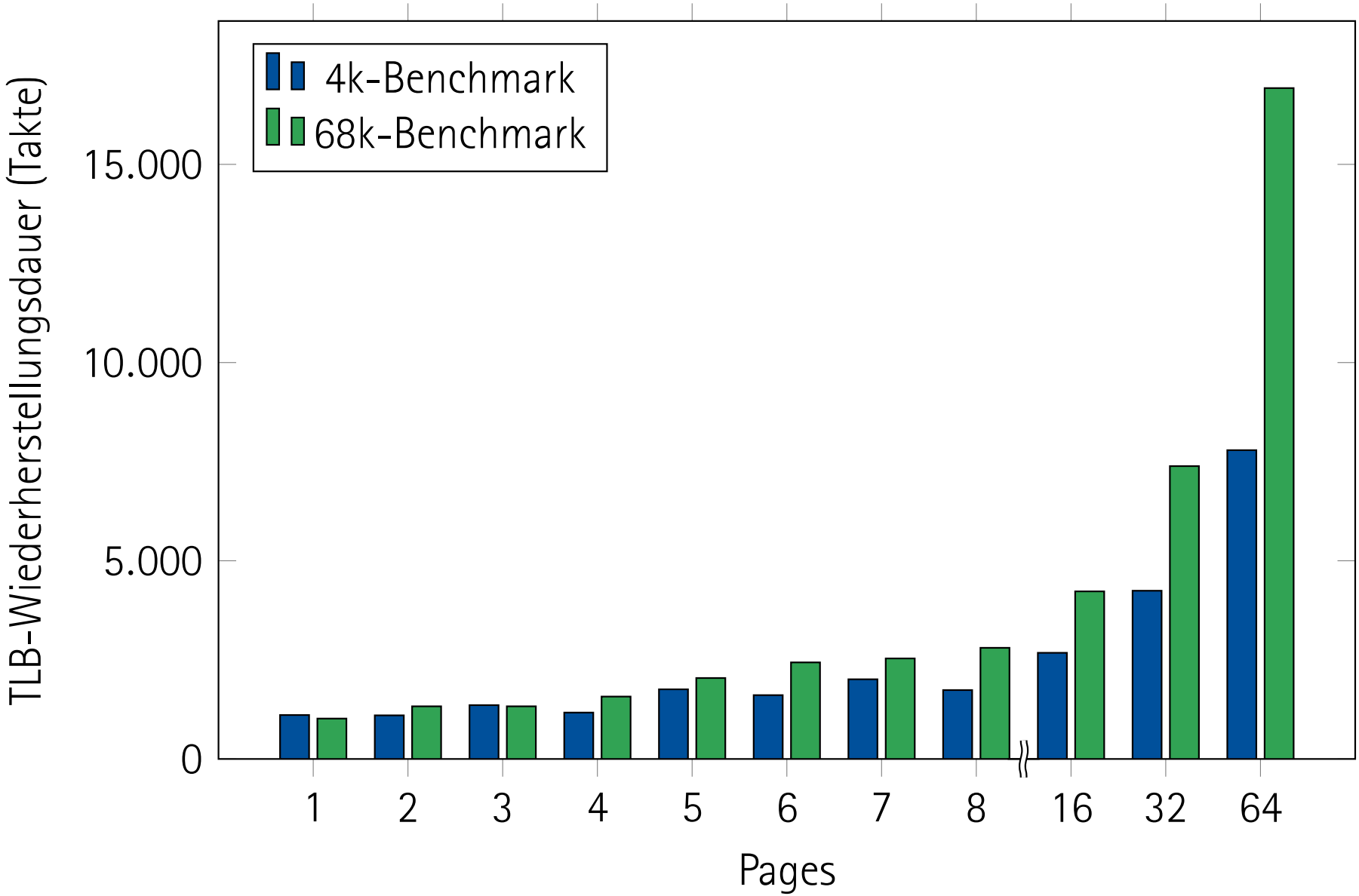
Zugriffszeitreduktion



TLB sichern


7.000
6.000
5.000
4.000
3.000
2.000
1.000
0
TLB Sicherungsdauer (Takte)
4k-Benchmark
68k-Benchmark
1
2
3
4
5
6
7
8
16
32
64
Pages

TLB wiederherstellen



Zwischenfazit

Ergebnis: TLB-Migration auf aktueller Hardware nicht rentabel

- Komplettes Auslesen des TLBs notwendig
- Max. 2 Speicherzugriffe im Hintergrund
- pld-Instruktion führt Table-Walk im Vordergrund aus

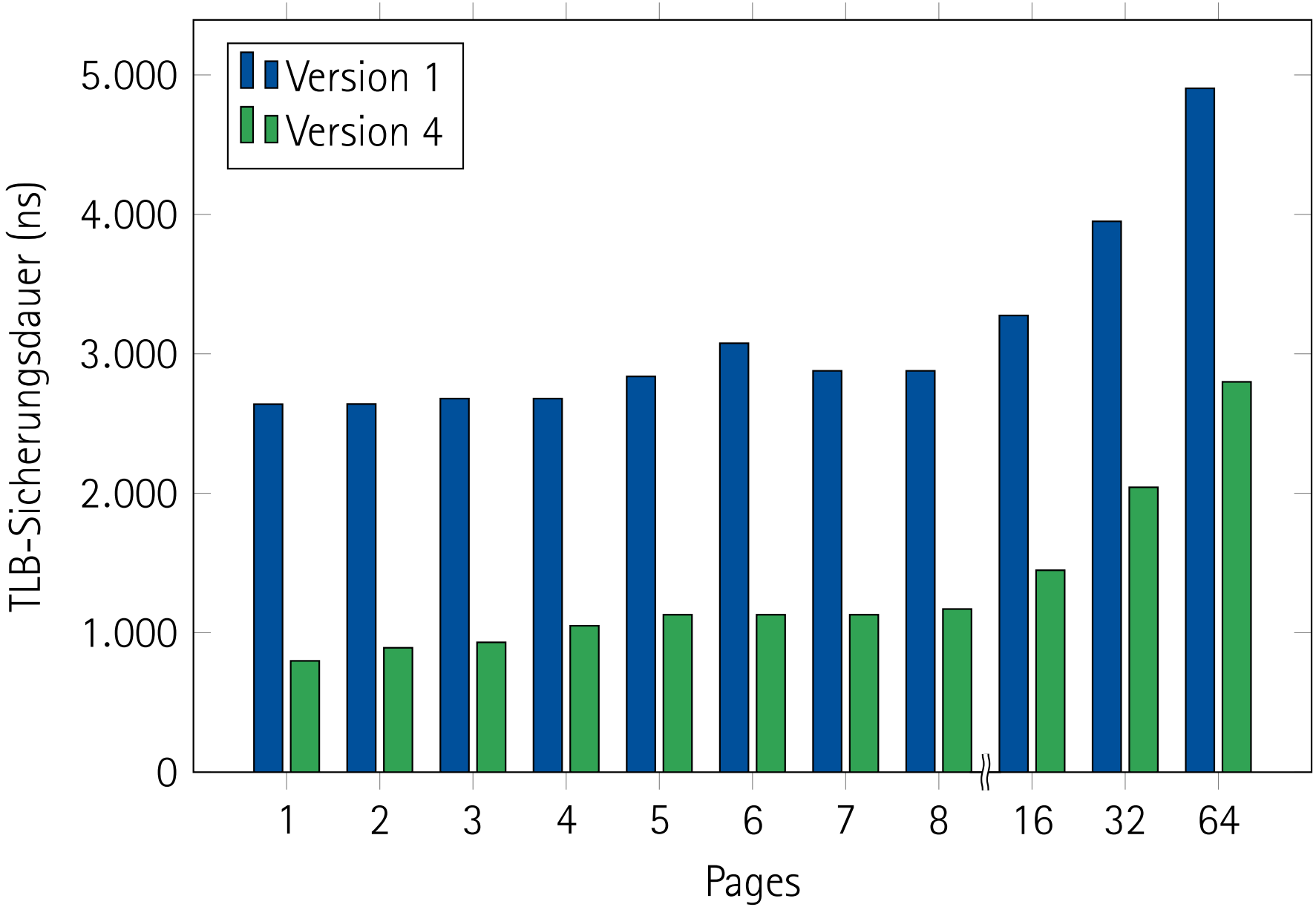
Hardwareanpassungen

Hardwareanpassungen

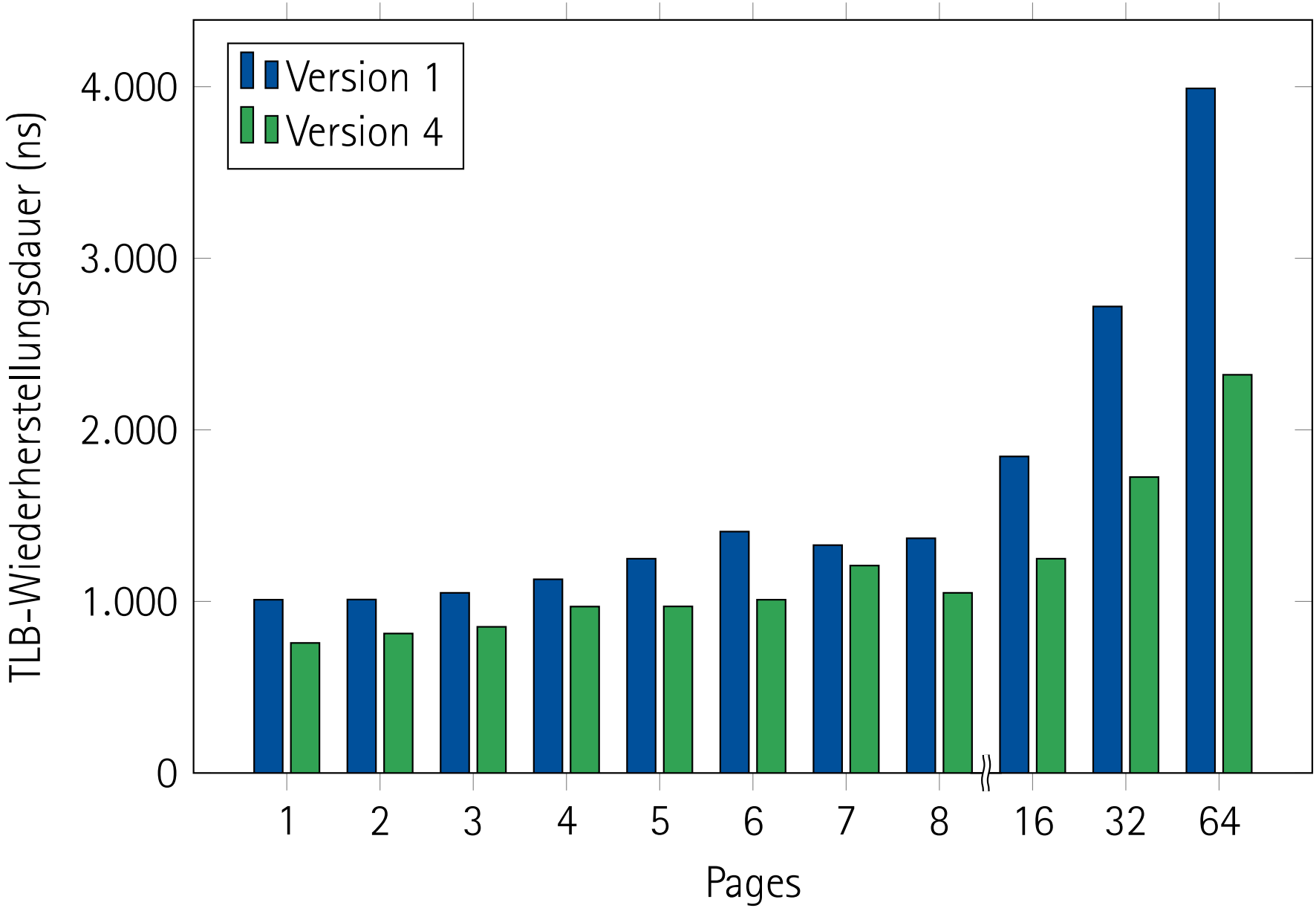
Hardwareanpassung mithilfe von gem5

- Schreibzugriff auf TLB-Interface (V1 + V4)
- Minimierung der zu transferierenden Daten (V4)
- Filtern der TLB-Einträge in Hardware (V4)
- Streaming der TLB-Einträge (V4)
- Byte-Packing des Datenstroms (V4)

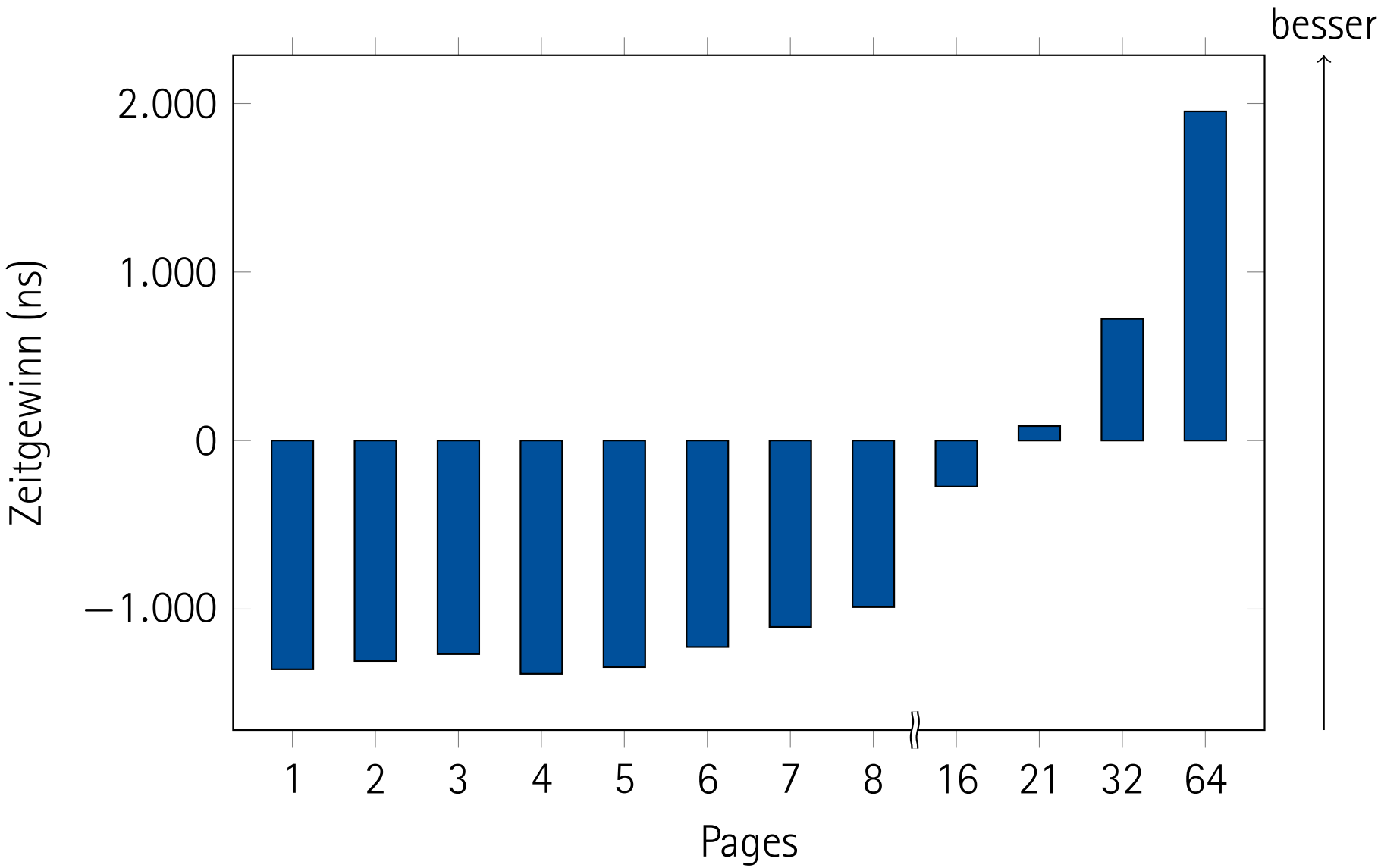
TLB sichern


Version 1
Version 4
TLB-Sicherungsdauer (ns)
5.000
4.000
3.000
2.000
1.000
0
1
2
3
4
5
6
7
8
16
32
64
Pages

TLB wiederherstellen

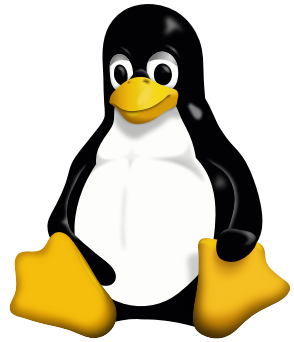


Zeitgewinn


besser
2.000
1.000
0
−1.000
Zeitgewinn (ns)
1
2
3
4
5
6
7
8
16
21
32
64
Pages

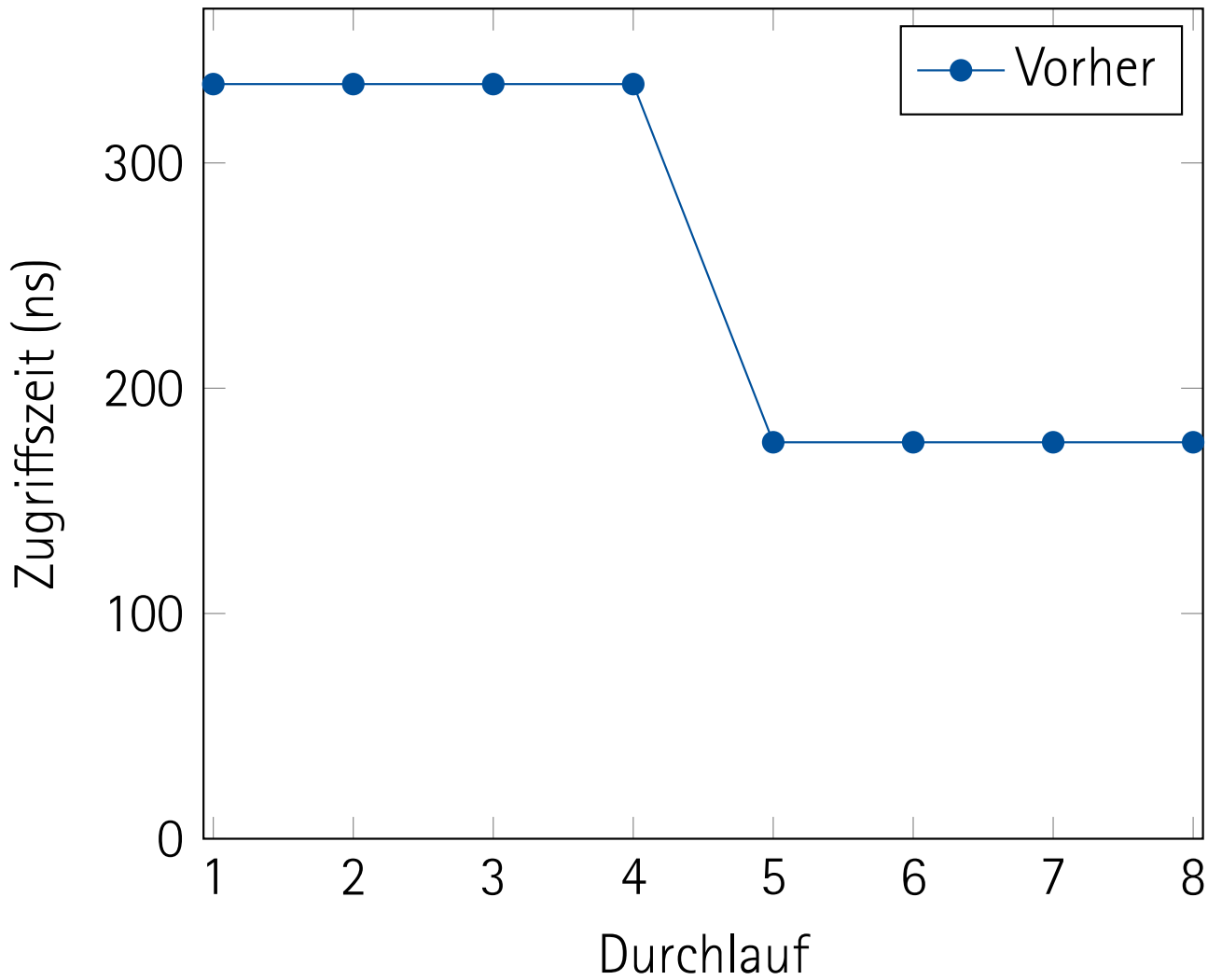
Linux-Integration

Linux-Integration

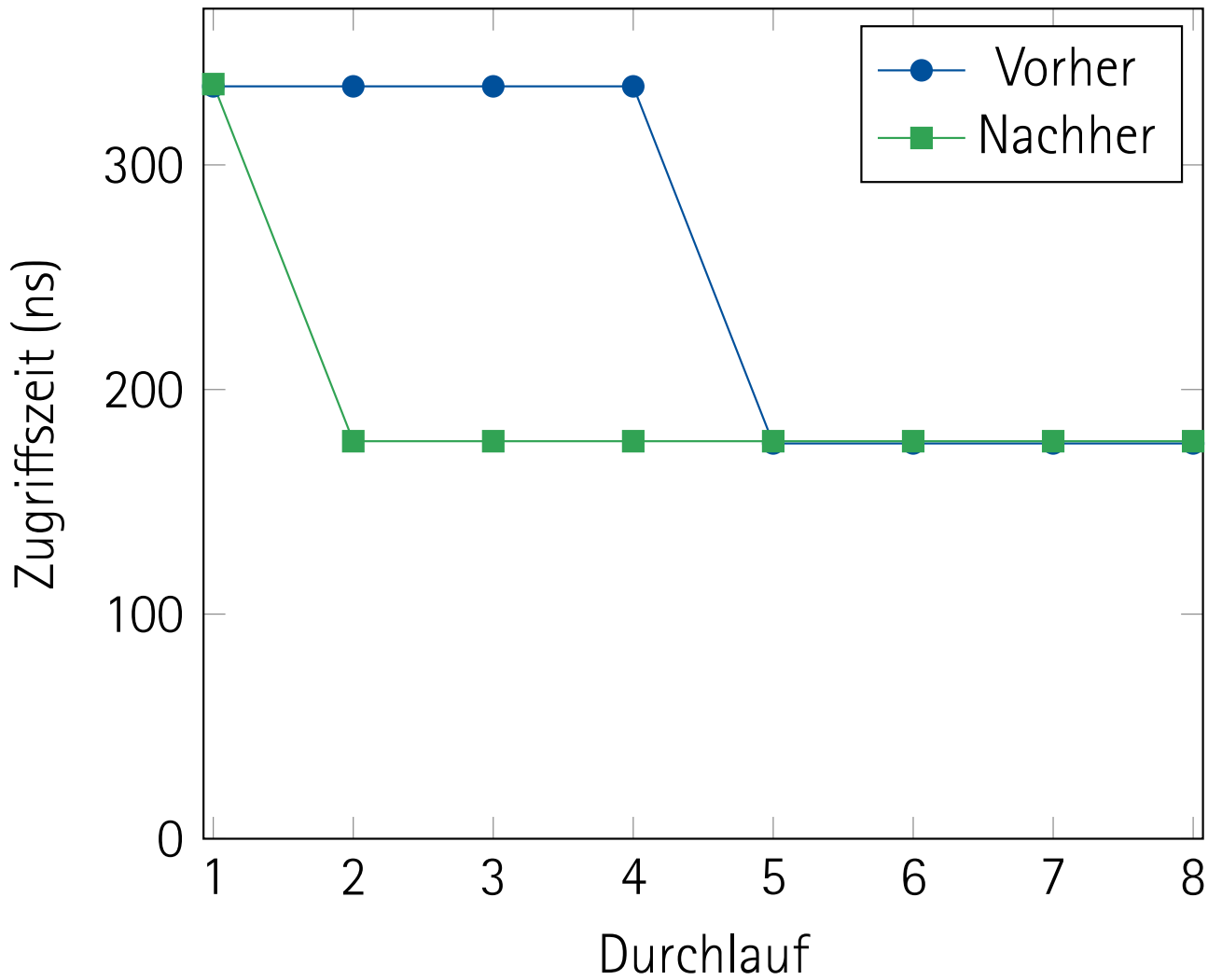


- Prozesse und Threads werden als Tasks abstrahiert
- Sichern des TLBs im `task_struct`
  - Annahme: liegt häufig im Cache
  - Begrenzt Speicherplatz für TLB-Einträge
- Sicherung und Wiederherstellung des TLBs in `move_queued_task()` und `finish_task_switch()`

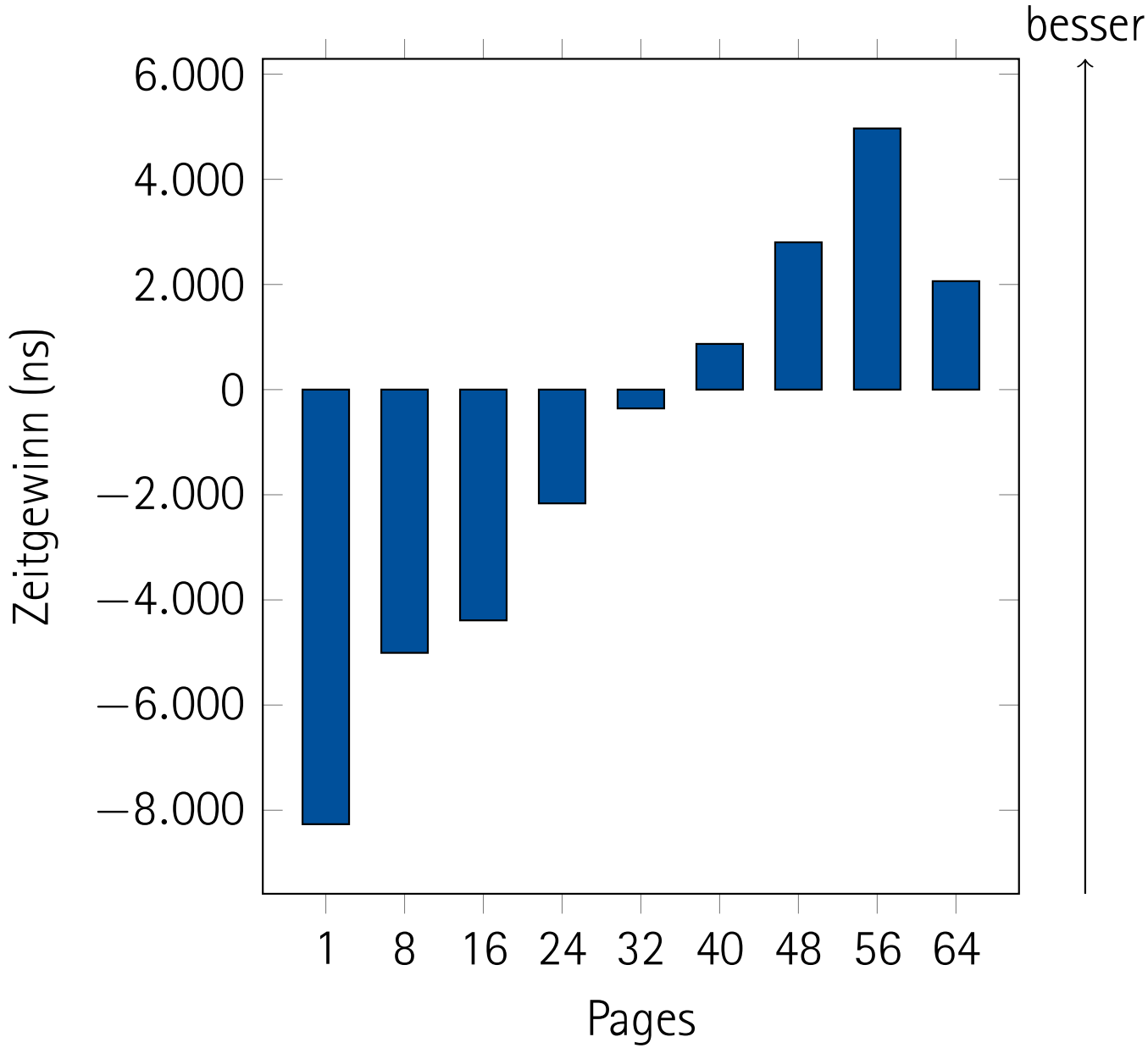
Linux-Benchmark



Linux-Benchmark



Linux-Benchmark (2)



Fazit

- TLB-Migration ohne Hardwareanpassungen nicht lohnenswert
- Geringfügige Hardwareanpassungen erlauben Verbesserungen in sehr speziellen Fällen
- Im Allgemeinen in aktueller Form zu langsam
- Ausblick
  - Betrachtung von realistischeren Arbeitslasten
  - Predictor zum Auswählen der zu migrierenden Pages
  - Implementierung für parallele Arbeitslasten