Sperren und deren Einfluss auf die Leistung von Mehrkern-Rechnern

Markus Büttner

Friedrich-Alexander Universität Erlangen-Nürnberg

Agenda

Motivation

Sperren

Testumgebung

LiTL

Messungen

Motivation

Motivation

- Sperren (noch) unvermeidbar

Motivation

- Sperren (noch) unvermeidbar
- Angebot an Leistungsstudien nicht zufriedenstellend

Motivation

- Sperren (noch) unvermeidbar
- Angebot an Leistungsstudien nicht zufriedenstellend
  - unzureichende Testmethoden

Motivation

- Sperren (noch) unvermeidbar
- Angebot an Leistungsstudien nicht zufriedenstellend
  - unzureichende Testmethoden
    - Microbenchmarks

Motivation

- Sperren (noch) unvermeidbar
- Angebot an Leistungsstudien nicht zufriedenstellend
  - unzureichende Testmethoden
    - Microbenchmarks
    - Test unter Optimalbedingungen

Motivation

- Sperren (noch) unvermeidbar
- Angebot an Leistungsstudien nicht zufriedenstellend
  - unzureichende Testmethoden
    - Microbenchmarks
    - Test unter Optimalbedingungen
  - Fokus auf andere Aspekte des Mehrkernbetriebs

Motivation

- Sperren (noch) unvermeidbar
- Angebot an Leistungsstudien nicht zufriedenstellend
  - unzureichende Testmethoden
    - Microbenchmarks
    - Test unter Optimalbedingungen
  - Fokus auf andere Aspekte des Mehrkernbetriebs
  - fehlende Breite an verschiedenen Algorithmen

Motivation

- Sperren (noch) unvermeidbar
- Angebot an Leistungsstudien nicht zufriedenstellend
  - unzureichende Testmethoden
    - Microbenchmarks
    - Test unter Optimalbedingungen
  - Fokus auf andere Aspekte des Mehrkernbetriebs
  - fehlende Breite an verschiedenen Algorithmen
- Multicore Locks: The Case Is Not Closed Yet
  - 27 Algorithmen
  - 35 Anwendungen
  - 3 Hardwarekonfigurationen

Agenda

Motivation

Sperren

Testumgebung

LiTL

Messungen

Sperren

Sperren

- ermöglichen gegenseitigen Ausschluss

Sperren

- ermöglichen gegenseitigen Ausschluss
- können Wettstreit verursachen

Sperren

- ermöglichen gegenseitigen Ausschluss
- können Wettstreit verursachen
- Einteilung in mehrere Klassen möglich

Klassifizierung

Klassifizierung

- Algorithmus

Klassifizierung

- Algorithmus
  - flach
    - TTAS

Klassifizierung

- Algorithmus
  - flach
    - TTAS

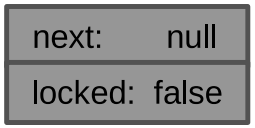
```
1  void lock(Lock *lock) {
2    do {
3      while(occupied(lock));
4    } while(!testAndLock(lock))
5  }
```

Klassifizierung

- Algorithmus
  - flach
    - TTAS
  - warteschlangenbasiert
    - MCS

Klassifizierung

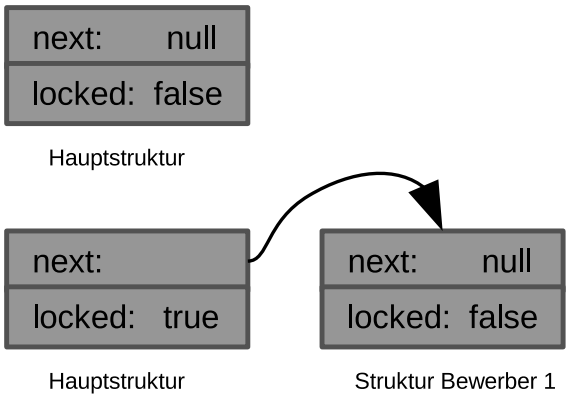
- Algorithmus
  - flach
    - TTAS
  - warteschlangenbasiert
    - MCS



Hauptstruktur

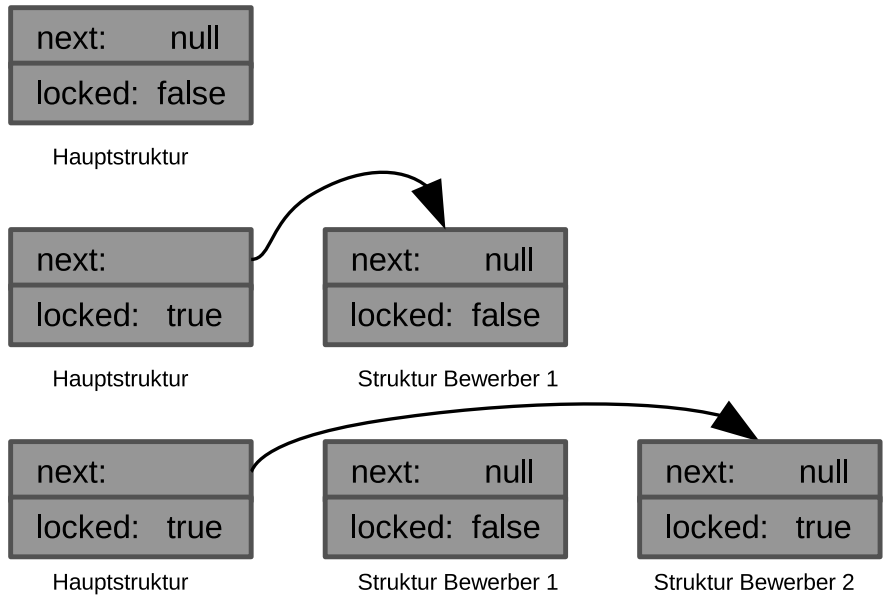
Klassifizierung

- Algorithmus
  - flach
    - TTAS
  - warteschlangenbasiert
    - MCS



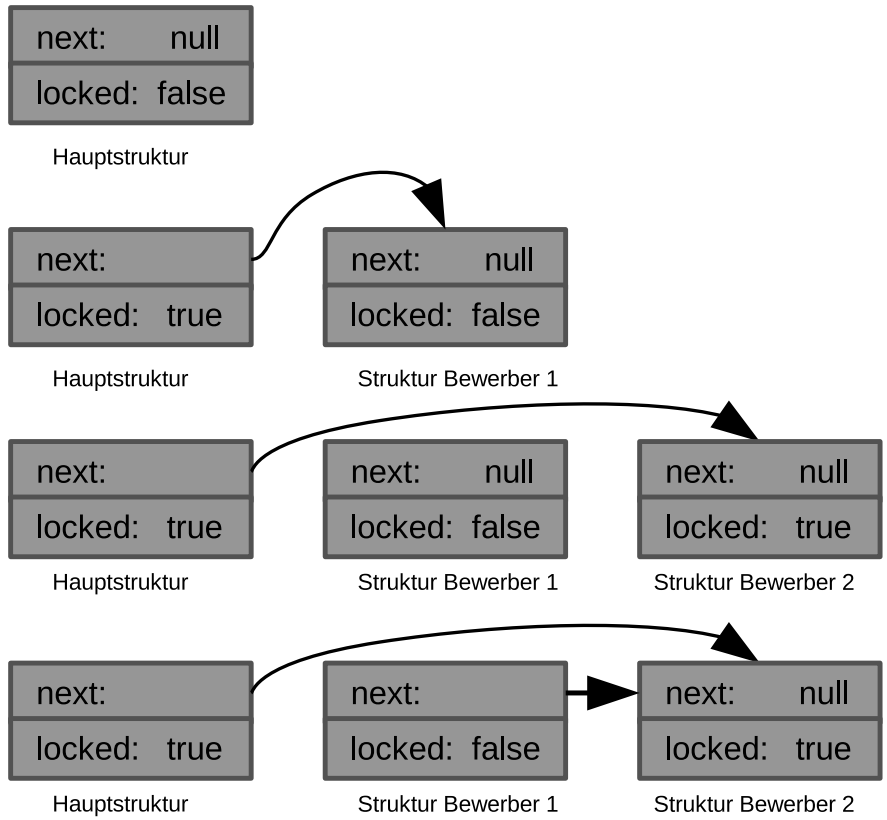
Klassifizierung

- Algorithmus
  - flach
    - TTAS
  - warteschlangenbasiert
    - MCS



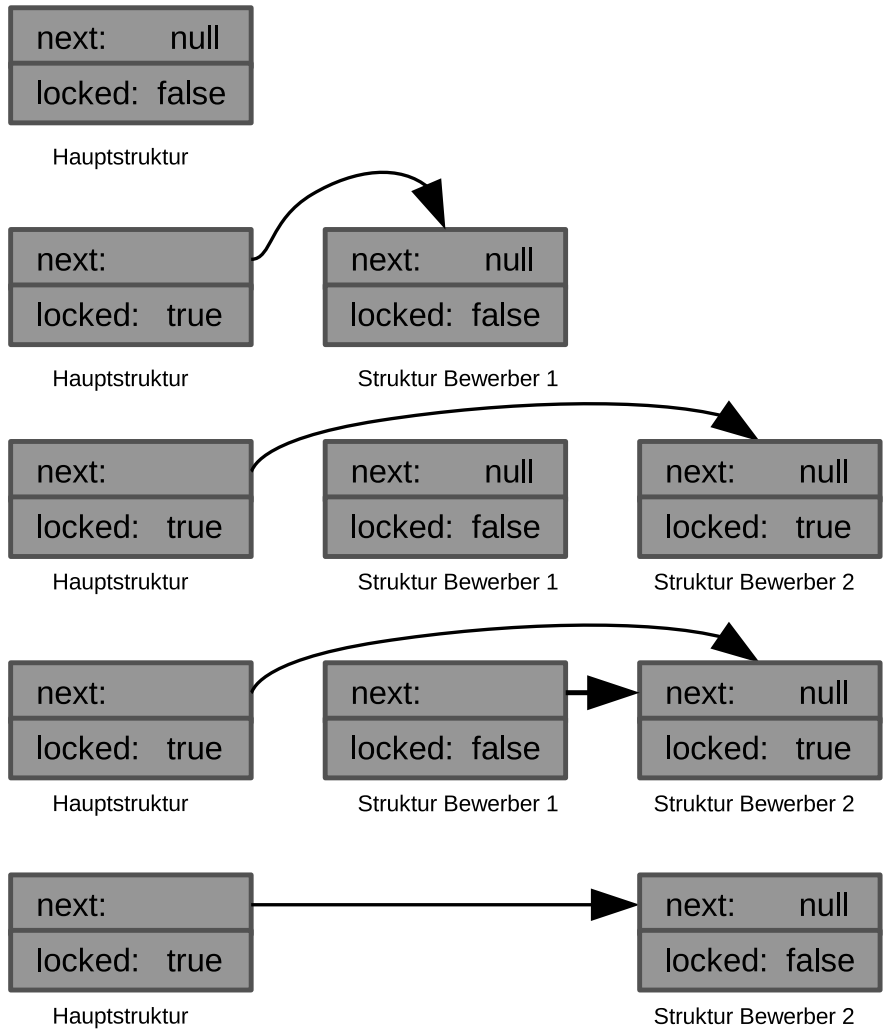
Klassifizierung

- Algorithmus
  - flach
    - TTAS
  - warteschlangenbasiert
    - MCS



Klassifizierung

- Algorithmus
  - flach
    - TTAS
  - warteschlangenbasiert
    - MCS

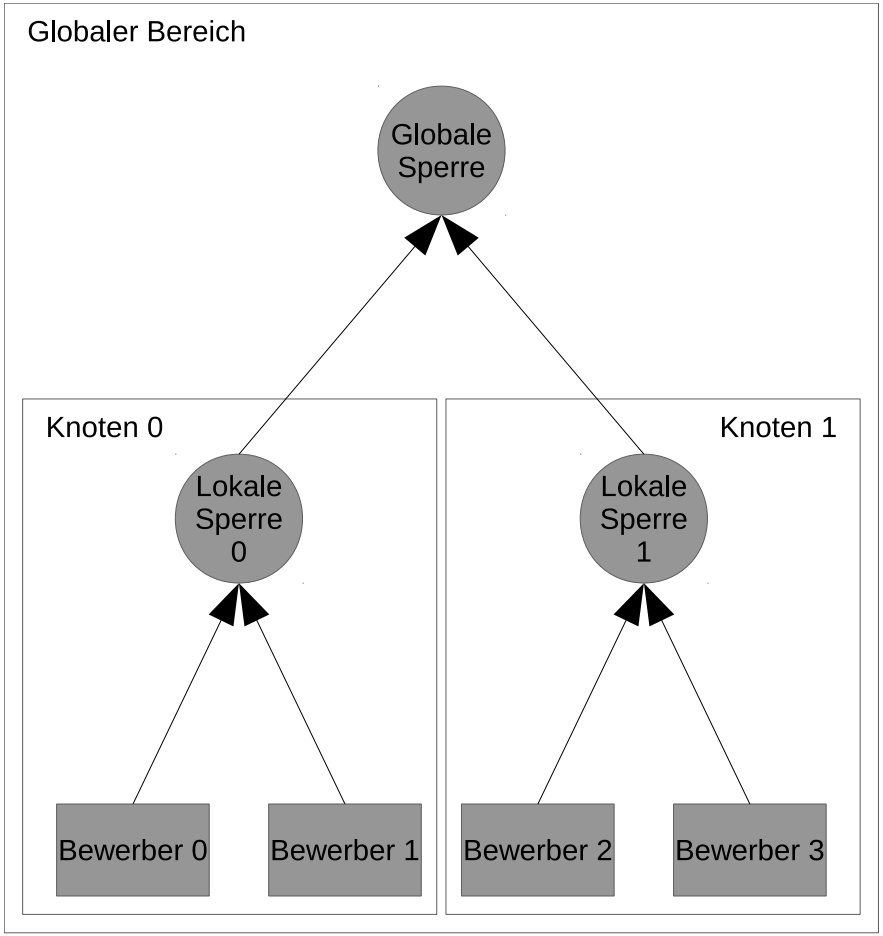


Klassifizierung

- Algorithmus
  - flach
    - TTAS
  - warteschlangenbasiert
    - MCS
  - hierarchisch
    - HMCS

Klassifizierung

- Algorithmus
  - flach
    - TTAS
  - warteschlangenbasiert
    - MCS
  - hierarchisch
    - HMCS



Klassifizierung

- Algorithmus
  - flach
    - TTAS
  - warteschlangenbasiert
    - MCS
  - hierarchisch
    - HMCS
  - laskontrollierend
    - MCS-TP

Klassifizierung

- Algorithmus
  - flach
    - TTAS
  - warteschlangenbasiert
    - MCS
  - hierarchisch
    - HMCS
  - laskontrollierend
    - MCS-TP
  - delegationsbasiert
    - RCL

Klassifizierung

- Algorithmus
  - flach
    - TTAS
  - warteschlangenbasiert
    - MCS
  - hierarchisch
    - HMCS
  - laskontrollierend
    - MCS-TP
  - delegationsbasiert
    - RCL
- Wartestrategie
  - Umlaufsperren
  - blockierende Sperren
  - hybrider Ansatz: STP

Auswahl

Auswahl

- keine eingeschränkt portablen Sperren

Auswahl

- keine eingeschränkt portablen Sperren
  - umfangreiche Anforderungen an Betriebssystem oder Anwendung

Auswahl

- keine eingeschränkt portablen Sperren
  - umfangreiche Anforderungen an Betriebssystem oder Anwendung
  - aufwändige Modifikationen im Quellcode

Auswahl

- keine eingeschränkt portablen Sperren
  - umfangreiche Anforderungen an Betriebssystem oder Anwendung
  - aufwändige Modifikationen im Quellcode
  - aufwändige Feineinstellungen

Auswahl

- keine eingeschränkt portablen Sperren
  - umfangreiche Anforderungen an Betriebssystem oder Anwendung
  - aufwändige Modifikationen im Quellcode
  - aufwändige Feineinstellungen
- keine delegationsbasierten Sperren

Auswahl

- keine eingeschränkt portablen Sperren
  - umfangreiche Anforderungen an Betriebssystem oder Anwendung
  - aufwändige Modifikationen im Quellcode
  - aufwändige Feineinstellungen
- keine delegationsbasierten Sperren
- jede weitere Klasse vertreten

Auswahl

- keine eingeschränkt portablen Sperren
  - umfangreiche Anforderungen an Betriebssystem oder Anwendung
  - aufwändige Modifikationen im Quellcode
  - aufwändige Feineinstellungen
- keine delegationsbasierten Sperren
- jede weitere Klasse vertreten
- einzelne Sperren mit verschiedenen Wartestrategien oder Optimierungen mehrfach vertreten

Agenda

Motivation

Sperren

Testumgebung

LiTL

Messungen

Testumgebung

Testumgebung

Hardware

Testumgebung

Hardware

- 3 Hardwarekonfigurationen
  - AMD-basiert 64 und 48 Kerne
  - Intel-basiert 48 Kerne

Testumgebung

Hardware

- 3 Hardwarekonfigurationen
  - AMD-basiert 64 und 48 Kerne
  - Intel-basiert 48 Kerne
- Messungen mit allen Konfigurationen durchgeführt

Testumgebung

Hardware

- 3 Hardwarekonfigurationen
  - AMD-basiert 64 und 48 Kerne
  - Intel-basiert 48 Kerne
- Messungen mit allen Konfigurationen durchgeführt
- Fokus auf AMD mit 64 Kernen

Testumgebung

Hardware

- 3 Hardwarekonfigurationen
  - AMD-basiert 64 und 48 Kerne
  - Intel-basiert 48 Kerne
- Messungen mit allen Konfigurationen durchgeführt
- Fokus auf AMD mit 64 Kernen

Software

Testumgebung

Hardware

- 3 Hardwarekonfigurationen
  - AMD-basiert 64 und 48 Kerne
  - Intel-basiert 48 Kerne
- Messungen mit allen Konfigurationen durchgeführt
- Fokus auf AMD mit 64 Kernen

Software

- Ubuntu 12.04

Testumgebung

Hardware

- 3 Hardwarekonfigurationen
  - AMD-basiert 64 und 48 Kerne
  - Intel-basiert 48 Kerne
- Messungen mit allen Konfigurationen durchgeführt
- Fokus auf AMD mit 64 Kernen

Software

- Ubuntu 12.04
- Linux Kernel 3.17.6

Testumgebung

Hardware

- 3 Hardwarekonfigurationen
  - AMD-basiert 64 und 48 Kerne
  - Intel-basiert 48 Kerne
- Messungen mit allen Konfigurationen durchgeführt
- Fokus auf AMD mit 64 Kernen

Software

- Ubuntu 12.04
- Linux Kernel 3.17.6
- Anwendungen für realistische Lastszenarien
  - PARSEC
  - Phoenix 2
  - SPLASH-2
  - MySQL
  - SSL-Proxy

Agenda

Motivation

Sperren

Testumgebung

LiTL

Messungen

LiTL

LiTL

- Möglichkeit zum einfachen Austausch von Sperren nötig

LiTL

- Möglichkeit zum einfachen Austausch von Sperren nötig
- Änderungen am Quelltext der Anwendungen vermeiden

LiTL

- Möglichkeit zum einfachen Austausch von Sperren nötig
- Änderungen am Quelltext der Anwendungen vermeiden
- zusätzlicher Rechenaufwand muss gering bleiben

LiTL

- Möglichkeit zum einfachen Austausch von Sperren nötig
- Änderungen am Quelltext der Anwendungen vermeiden
- zusätzlicher Rechenaufwand muss gering bleiben
- Zusatzfunktionen wie Zustandsvariablen müssen erhalten bleiben

Umsetzung

```
1  pthread_mutex_lock(pthread_mutex_t *m) {
2          optimized_mutex_t *om = get_optimized_mutex(m);
3          if (om == null) {
4                  om = create_and_store_optimized_mutex(m);
5          }
6          optimized_mutex_lock(om);
7          real_pthread_mutex_lock(m);
8  }
9
10 pthread_mutex_unlock(pthread_mutex_t *m) {
11         optimized_mutex_t *om = get_optmized_mutex(m);
12         optimized_mutex_unlock(om);
13         real_pthread_mutex_unlock(m);
14 }
15
16 pthread_cond_wait(pthread_cond_t *c, pthread_mutex_t *m) {
17         optimized_mutex_t *om = get_optimized_mutex(m);
18         optimized_mutex_unlock(om);
19         real_pthread_cond_wait(c, m);
20         real_pthread_mutex_unlock(m);
21         optimized_mutex_lock(om);
22         real_pthread_mutex_lock(m);
23 }
```

Umsetzung

```
1  pthread_mutex_lock(pthread_mutex_t *m) {
2          optimized_mutex_t *om = get_optimized_mutex(m);
3          if (om == null) {
4                  om = create_and_store_optimized_mutex(m);
5          }
6          optimized_mutex_lock(om);
7          real_pthread_mutex_lock(m);
8  }
9
10 pthread_mutex_unlock(pthread_mutex_t *m) {
11         optimized_mutex_t *om = get_optmized_mutex(m);
12         optimized_mutex_unlock(om);
13         real_pthread_mutex_unlock(m);
14 }
15
16 pthread_cond_wait(pthread_cond_t *c, pthread_mutex_t *m) {
17         optimized_mutex_t *om = get_optimized_mutex(m);
18         optimized_mutex_unlock(om);
19         real_pthread_cond_wait(c, m);
20         real_pthread_mutex_unlock(m);
21         optimized_mutex_lock(om);
22         real_pthread_mutex_lock(m);
23 }
```

Umsetzung

```
1  pthread_mutex_lock(pthread_mutex_t *m) {
2          optimized_mutex_t *om = get_optimized_mutex(m);
3          if (om == null) {
4                  om = create_and_store_optimized_mutex(m);
5          }
6          optimized_mutex_lock(om);
7          real_pthread_mutex_lock(m);
8  }
9
10 pthread_mutex_unlock(pthread_mutex_t *m) {
11         optimized_mutex_t *om = get_optmized_mutex(m);
12         optimized_mutex_unlock(om);
13         real_pthread_mutex_unlock(m);
14 }
15
16 pthread_cond_wait(pthread_cond_t *c, pthread_mutex_t *m) {
17         optimized_mutex_t *om = get_optimized_mutex(m);
18         optimized_mutex_unlock(om);
19         real_pthread_cond_wait(c, m);
20         real_pthread_mutex_unlock(m);
21         optimized_mutex_lock(om);
22         real_pthread_mutex_lock(m);
23 }
```

Umsetzung

```
1  pthread_mutex_lock(pthread_mutex_t *m) {
2          optimized_mutex_t *om = get_optimized_mutex(m);
3          if (om == null) {
4                  om = create_and_store_optimized_mutex(m);
5          }
6          optimized_mutex_lock(om);
7          real_pthread_mutex_lock(m);
8  }
9
10 pthread_mutex_unlock(pthread_mutex_t *m) {
11         optimized_mutex_t *om = get_optmized_mutex(m);
12         optimized_mutex_unlock(om);
13         real_pthread_mutex_unlock(m);
14 }
15
16 pthread_cond_wait(pthread_cond_t *c, pthread_mutex_t *m) {
17         optimized_mutex_t *om = get_optimized_mutex(m);
18         optimized_mutex_unlock(om);
19         real_pthread_cond_wait(c, m);
20         real_pthread_mutex_unlock(m);
21         optimized_mutex_lock(om);
22         real_pthread_mutex_lock(m);
23 }
```

Umsetzung

```
1  pthread_mutex_lock(pthread_mutex_t *m) {
2          optimized_mutex_t *om = get_optimized_mutex(m);
3          if (om == null) {
4                  om = create_and_store_optimized_mutex(m);
5          }
6          optimized_mutex_lock(om);
7          real_pthread_mutex_lock(m);
8  }
9
10 pthread_mutex_unlock(pthread_mutex_t *m) {
11         optimized_mutex_t *om = get_optmized_mutex(m);
12         optimized_mutex_unlock(om);
13         real_pthread_mutex_unlock(m);
14 }
15
16 pthread_cond_wait(pthread_cond_t *c, pthread_mutex_t *m) {
17         optimized_mutex_t *om = get_optimized_mutex(m);
18         optimized_mutex_unlock(om);
19         real_pthread_cond_wait(c, m);
20         real_pthread_mutex_unlock(m);
21         optimized_mutex_lock(om);
22         real_pthread_mutex_lock(m);
23 }
```

Verifizierung

Verifizierung

- Erzeugung besonders schwieriger Umstände

Verifizierung

- Erzeugung besonders schwieriger Umstände
  - Auswahl ungünstiger Anwendungen
    - viele Sperren
    - Zustandsvariablen

Verifizierung

- Erzeugung besonders schwieriger Umstände
  - Auswahl ungünstiger Anwendungen
    - viele Sperren
    - Zustandsvariablen
  - maximales Wettbewerbsniveau

Verifizierung

- Erzeugung besonders schwieriger Umstände
  - Auswahl ungünstiger Anwendungen
    - viele Sperren
    - Zustandsvariablen
  - maximales Wettbewerbsniveau
- Vergleich der Zwischenschaltung durch LiTL mit manueller Modifikation

Verifizierung

- Erzeugung besonders schwieriger Umstände
  - Auswahl ungünstiger Anwendungen
    - viele Sperren
    - Zustandsvariablen
  - maximales Wettbewerbsniveau
- Vergleich der Zwischenschaltung durch LiTL mit manueller Modifikation
  - durchschnittlich weniger als 5% Differenz in der Leistung

Verifizierung

- Erzeugung besonders schwieriger Umstände
  - Auswahl ungünstiger Anwendungen
    - viele Sperren
    - Zustandsvariablen
  - maximales Wettbewerbsniveau
- Vergleich der Zwischenschaltung durch LiTL mit manueller Modifikation
  - durchschnittlich weniger als 5% Differenz in der Leistung
  - schlechtere Leistung durch LiTL allerdings erkennbar

Agenda

Motivation

Sperren

Testumgebung

LiTL

Messungen

Messungen

Messungen

- Einheit der Messungen: *Operationen pro Zeiteinheit*

Messungen

- Einheit der Messungen: *Operationen pro Zeiteinheit*
- Durchschnitt aus 5 Messungen

Messungen

- Einheit der Messungen: *Operationen pro Zeiteinheit*
- Durchschnitt aus 5 Messungen
- Sperre wird im Vergleich ab 5% mehr Leistung als besser betrachtet

Messungen

- Einheit der Messungen: *Operationen pro Zeiteinheit*
- Durchschnitt aus 5 Messungen
- Sperre wird im Vergleich ab 5% mehr Leistung als besser betrachtet
- nicht kompatible Kombinationen aus Sperren und Anwendungen werden nicht betrachtet

Messungen

- Einheit der Messungen: *Operationen pro Zeiteinheit*
- Durchschnitt aus 5 Messungen
- Sperre wird im Vergleich ab 5% mehr Leistung als besser betrachtet
- nicht kompatible Kombinationen aus Sperren und Anwendungen werden nicht betrachtet
- auch Pthread-Sperren werden mit LiTL gemessen

Messungen

- Einheit der Messungen: *Operationen pro Zeiteinheit*
- Durchschnitt aus 5 Messungen
- Sperre wird im Vergleich ab 5% mehr Leistung als besser betrachtet
- nicht kompatible Kombinationen aus Sperren und Anwendungen werden nicht betrachtet
- auch Pthread-Sperren werden mit LiTL gemessen
- Anpassung des Wettbewerbsniveaus auf Knotenebene
  - A-64: 8 Kerne
  - A-48: 6 Kerne
  - I-48: 12 Kerne

Überlegungen

Überlegungen

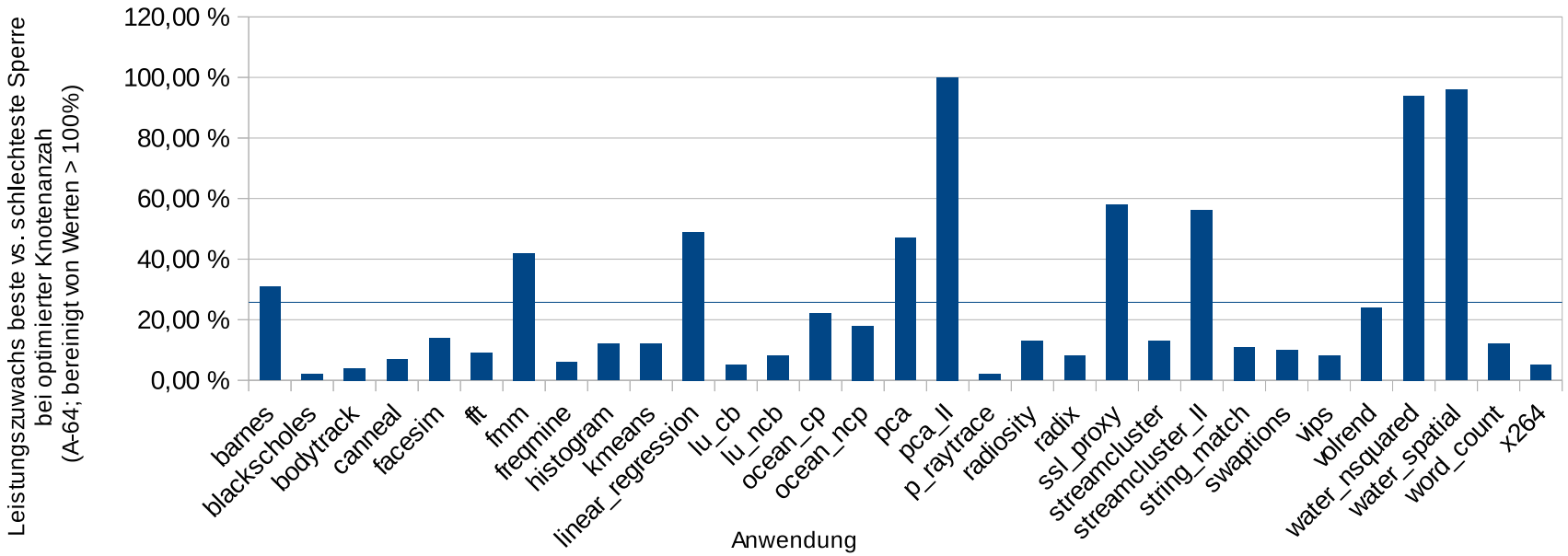
- relevante Anwendungen
  - beeinflussbar durch Wahl der Sperre
  - 10% relative Standardabweichung in der Leistung als Grenze
  - nur 23 der Anwendungen kommen für A-64 in Frage

Überlegungen

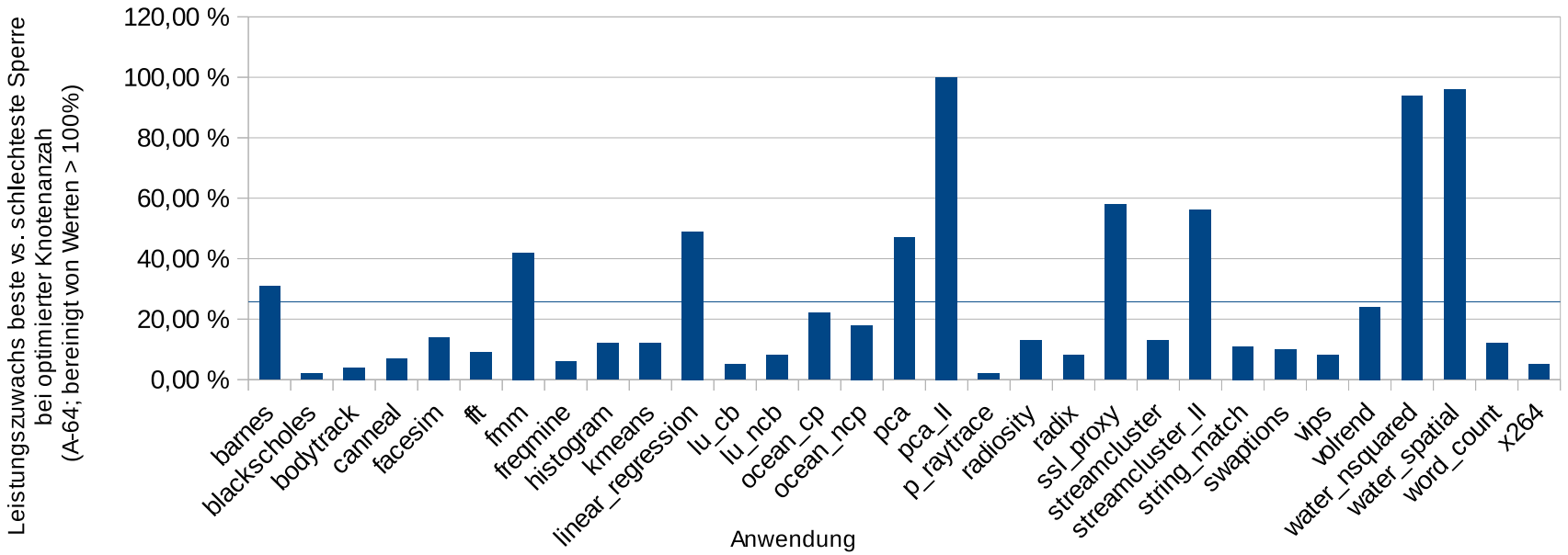
- relevante Anwendungen
  - beeinflussbar durch Wahl der Sperre
  - 10% relative Standardabweichung in der Leistung als Grenze
  - nur 23 der Anwendungen kommen für A-64 in Frage
- optimale Anzahl an verwendeten Knoten
  - Leistung skaliert nur begrenzt mit Knotenanzahl
  - Leistung mit einem oder allen Knoten nicht immer optimal
  - zusätzliche Messung mit optimierter Knotenanzahl

Ergebnisse

Ergebnisse

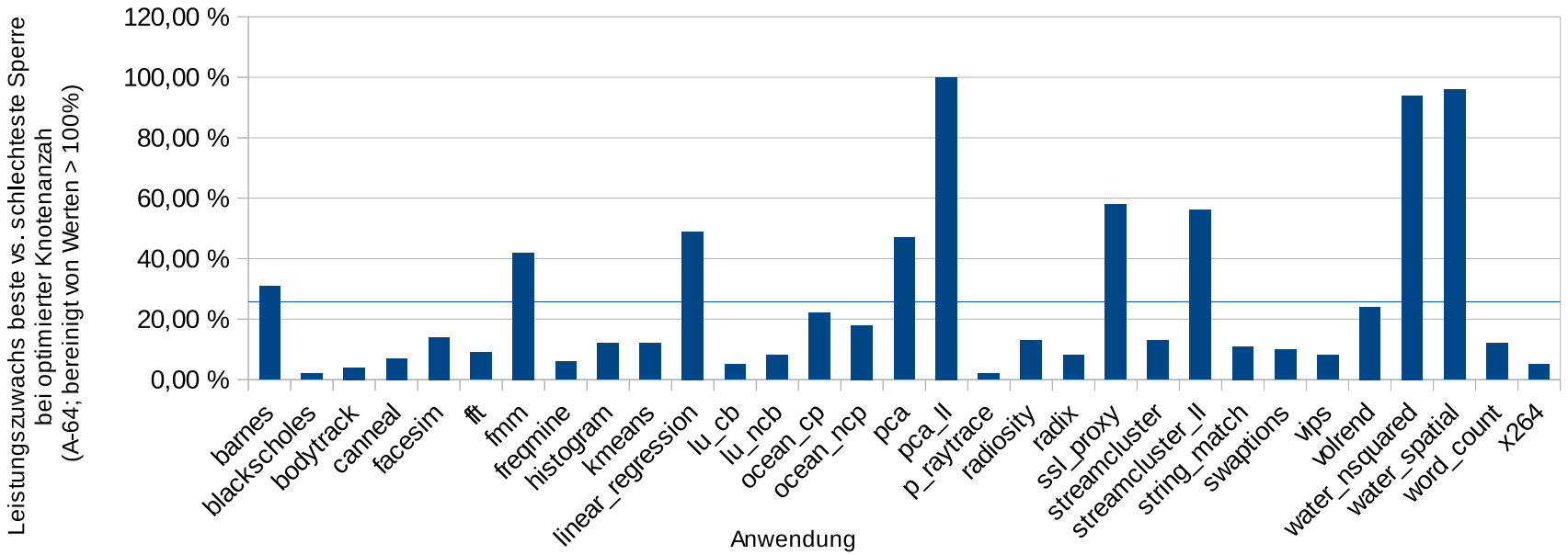

Leistungszuwachs beste vs. schlechteste Sperre
bei optimierter Knotenanzah
(A-64; bereinigt von Werten > 100%)
120,00 %
100,00 %
80,00 %
60,00 %
40,00 %
20,00 %
0,00 %
barnes
blackscholes
bodytrack
canneal
facesim
fft
fmm
freqmine
histogram
kmeans
linear_regression
lu_cb
lu_ncb
ocean_cp
ocean_ncp
pca
pca_ll
p_raytrace
radiosity
radix
ssl_proxy
streamcluster
streamcluster_ll
string_match
swaptions
vips
volrend
water_nsquared
water_spatial
word_count
x264
Anwendung

Ergebnisse



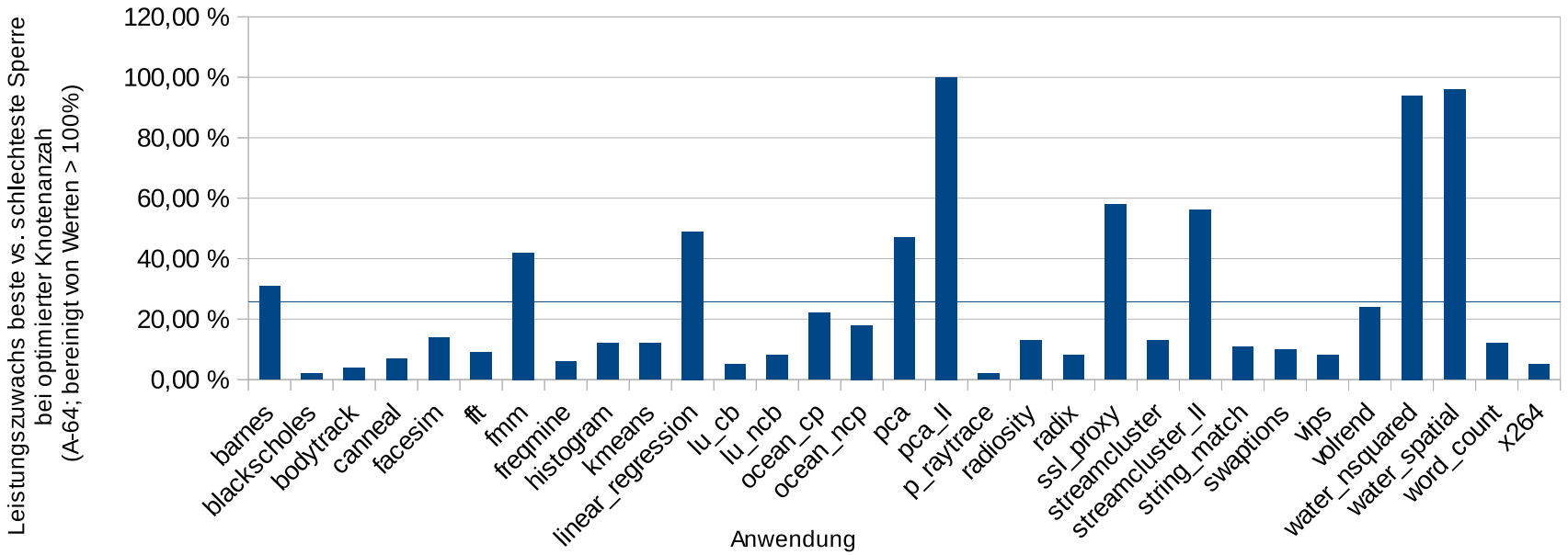
- Optimierungspotential

Ergebnisse



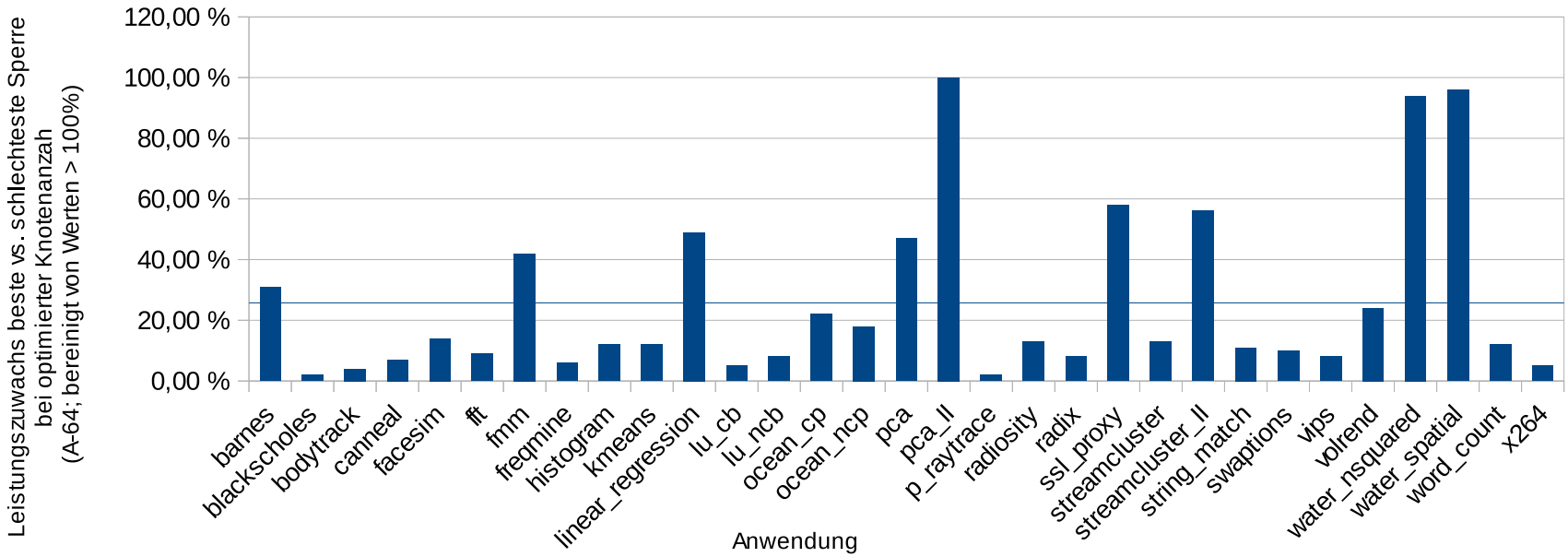
- Optimierungspotential
  - Verwendung der richtigen Sperre kann starken Leistungszuwachs bringen

Ergebnisse



- Optimierungspotential
  - Verwendung der richtigen Sperre kann starken Leistungszuwachs bringen
  - Messungen mit optimaler und maximaler Knotenanzahl besonders betroffen

Ergebnisse



Beobachtung

Ungefähr 60% der untersuchten Anwendungen werden stark von der verwendeten Sperre beeinflusst

Ergebnisse

- Beste Sperre

Ergebnisse

- Beste Sperre
  - AHMCS gehört zu den besten 5% der Sperren bei 52% der Anwendungen

Ergebnisse

- Beste Sperre
  - AHMCS gehört zu den besten 5% der Sperren bei 52% der Anwendungen
  - anderer Sperren erreichen bis zu 48%

Ergebnisse

- Beste Sperre
  - AHMCS gehört zu den besten 5% der Sperren bei 52% der Anwendungen
  - anderer Sperren erreichen bis zu 48%
  - auch einfache Sperren erzielen gute Leistungen

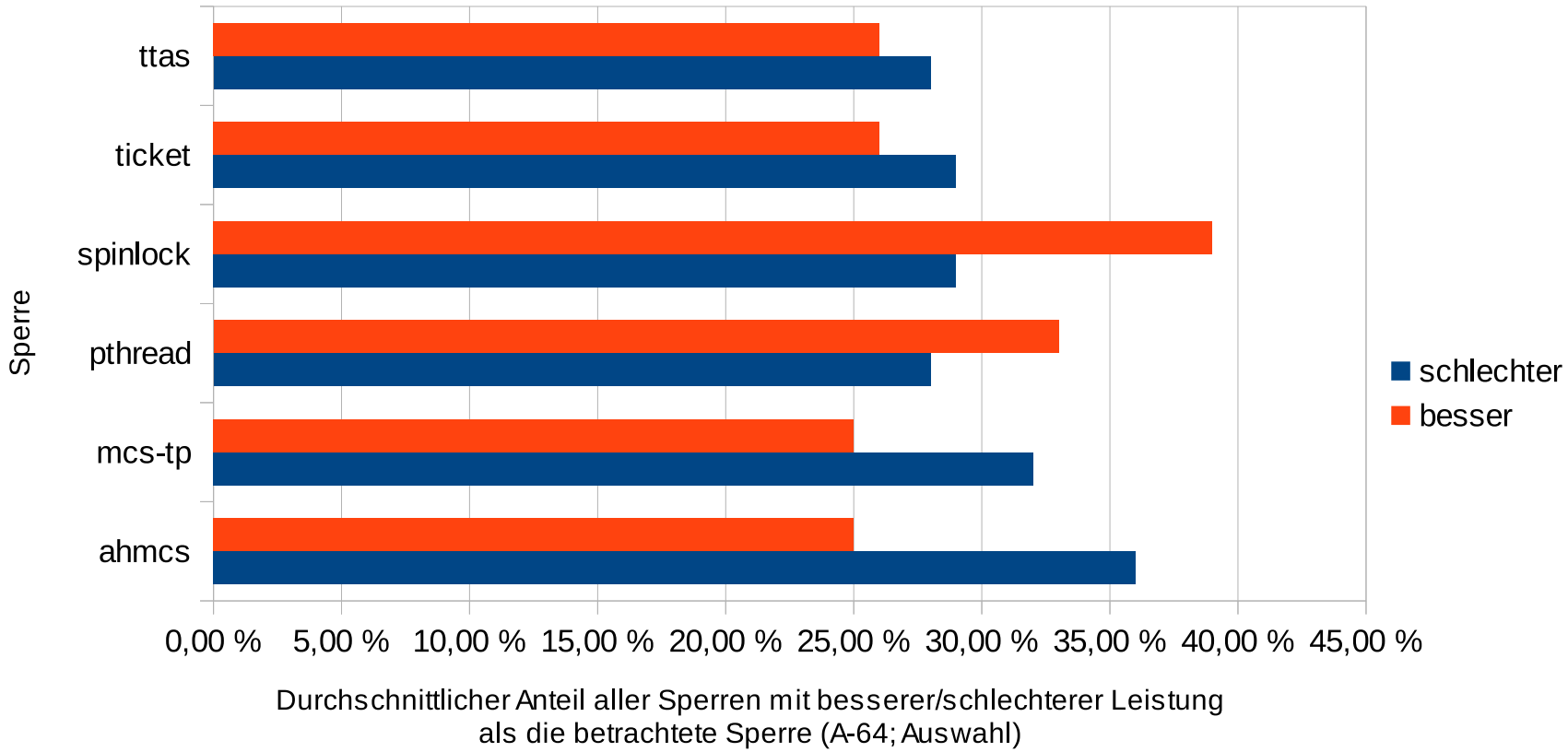
Ergebnisse

- Beste Sperre
  - AHMCS gehört zu den besten 5% der Sperren bei 52% der Anwendungen
  - anderer Sperren erreichen bis zu 48%
  - auch einfache Sperren erzielen gute Leistungen

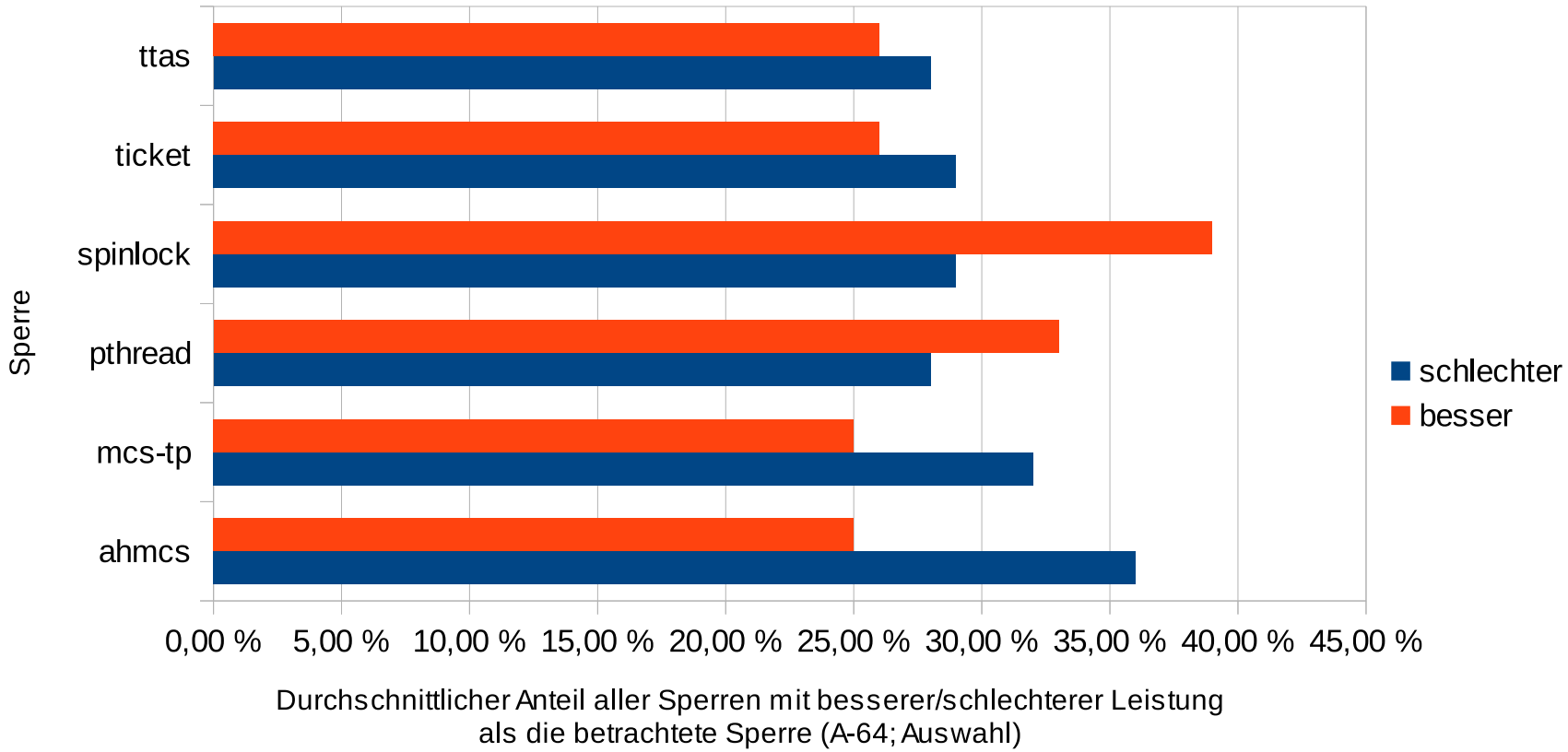
Beobachtung

Keine Sperre erbringt für alle Anwendungen die beste Leistung

Ergebnisse

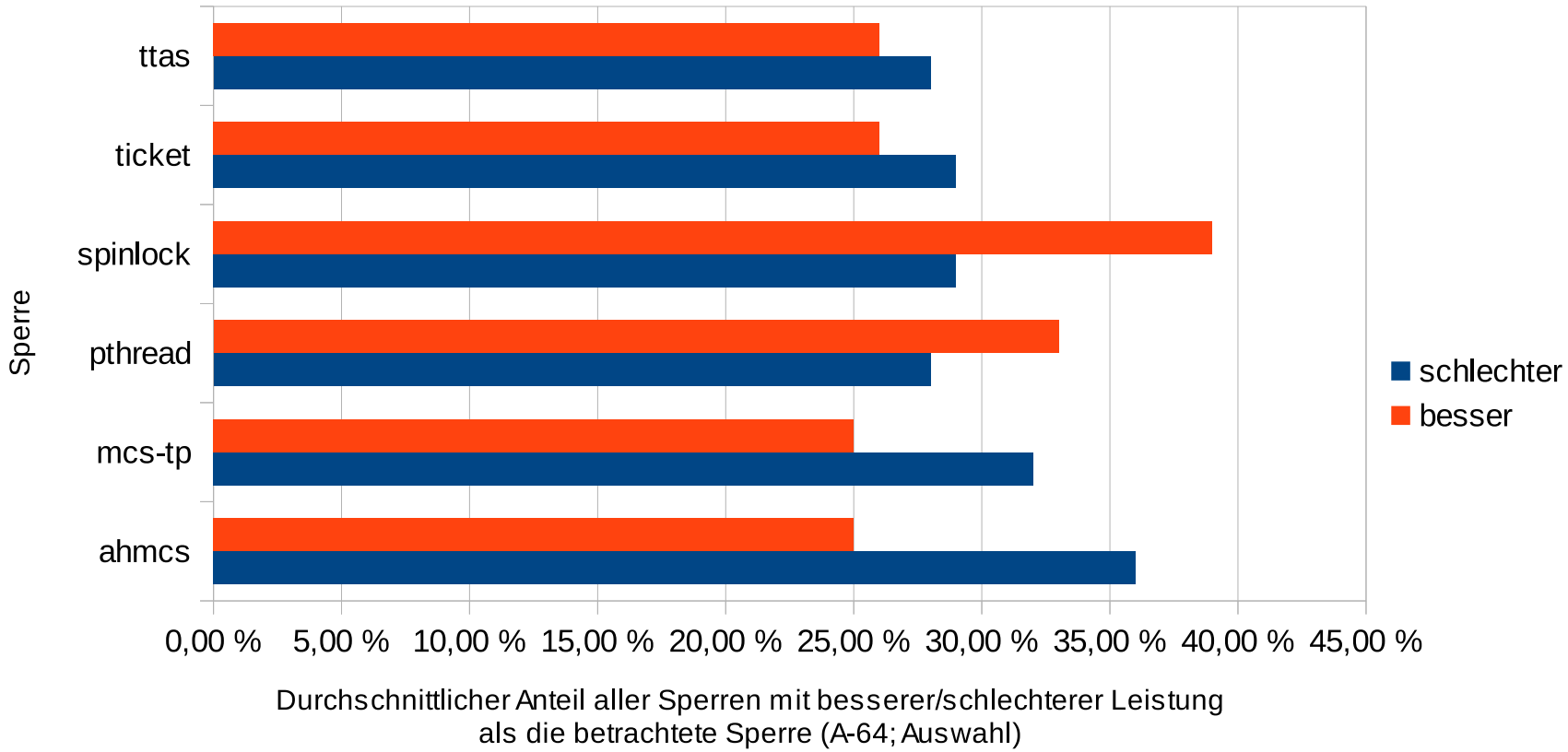

ttas
ticket
spinlock
pthread
mcs-tp
ahmcs
Sperre
0,00 %
5,00 %
10,00 %
15,00 %
20,00 %
25,00 %
30,00 %
35,00 %
40,00 %
45,00 %
schlechter
besser
Durchschnittlicher Anteil aller Sperren mit besserer/schlechterer Leistung
als die betrachtete Sperre (A-64; Auswahl)

Ergebnisse



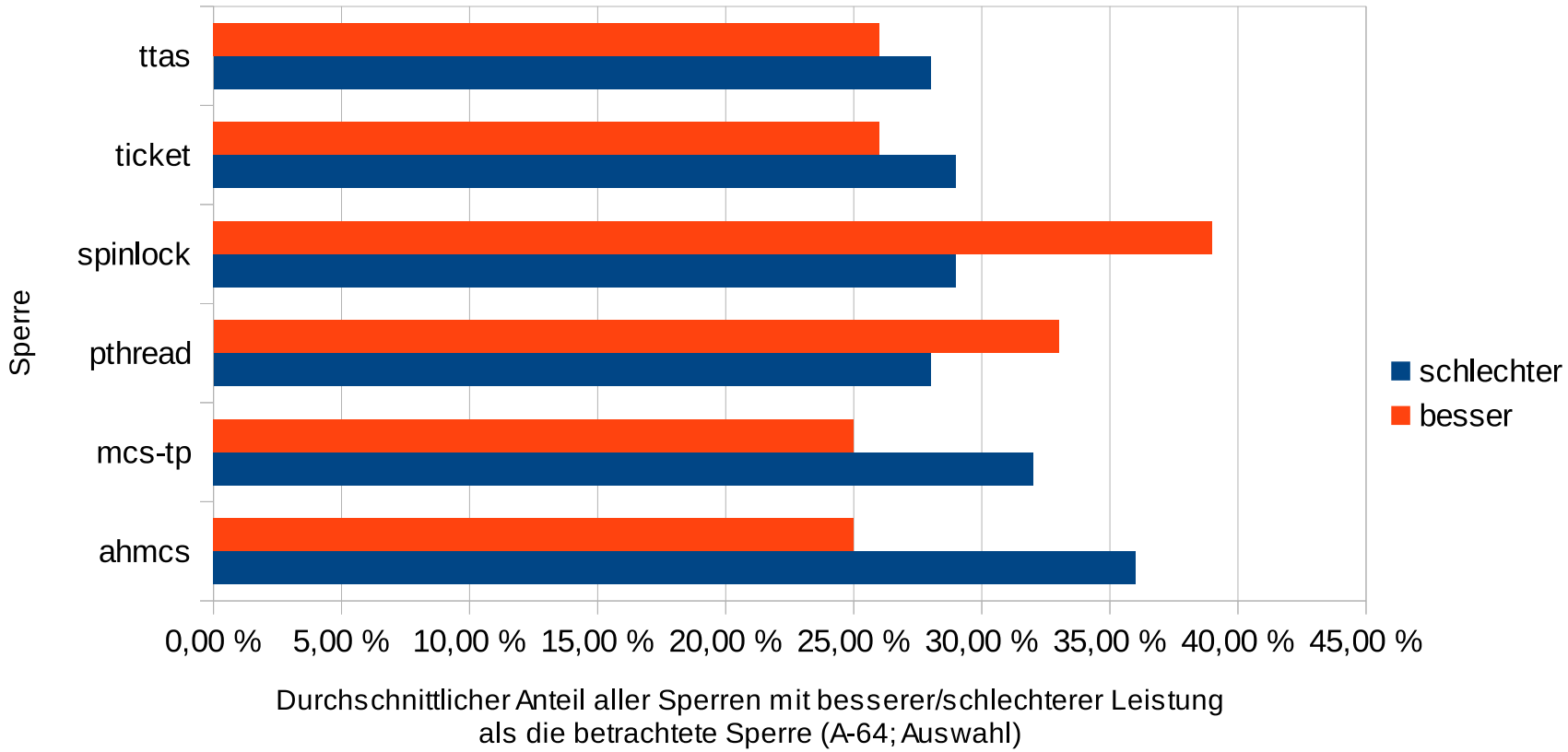
- Leistungsrangfolge

Ergebnisse



- Leistungsrangfolge
  - Anteil der Siege und Niederlagen im paarweisen Vergleich ähnlich

Ergebnisse



Beobachtung

Es ist keine eindeutige, globale Rangfolge bezüglich der Leistungsfähigkeit der Sperren erkennbar

Ergebnisse

- Negativer Einfluss von Sperren

Ergebnisse

- Negativer Einfluss von Sperren
  - keine Sperre erzielt nur positive Ergebnisse

Ergebnisse

- Negativer Einfluss von Sperren
  - keine Sperre erzielt nur positive Ergebnisse
  - AHMCS auf A-64 schädlich für
    - 24% der Anwendungen bei optimierter Knotenanzahl
    - 62% der Anwendungen bei maximaler Knotenanzahl

Ergebnisse

- Negativer Einfluss von Sperren
  - keine Sperre erzielt nur positive Ergebnisse
  - AHMCS auf A-64 schädlich für
    - 24% der Anwendungen bei optimierter Knotenanzahl
    - 62% der Anwendungen bei maximaler Knotenanzahl
  - gleiche Problematik bei allen anderen Sperren

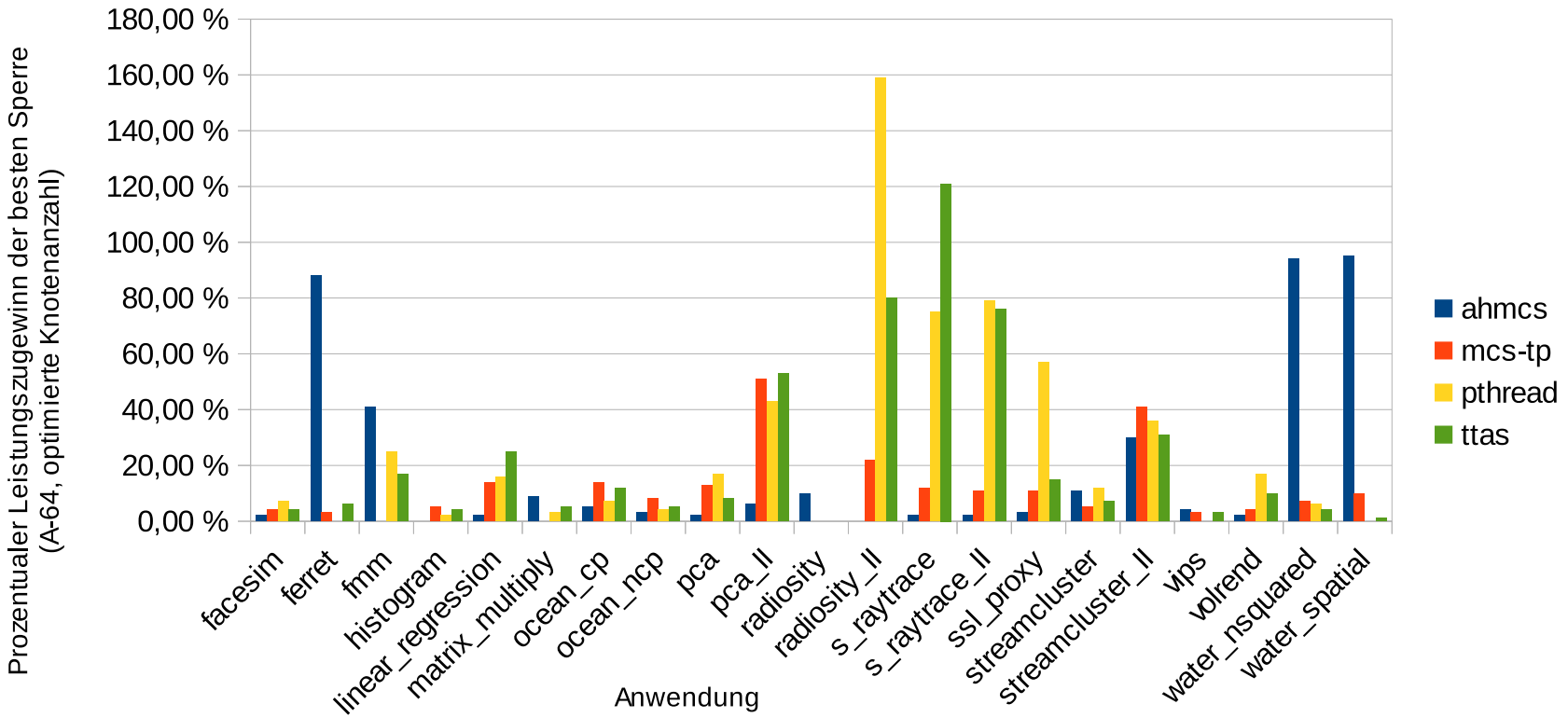
Ergebnisse

- Negativer Einfluss von Sperren
  - keine Sperre erzielt nur positive Ergebnisse
  - AHMCS auf A-64 schädlich für
    - 24% der Anwendungen bei optimierter Knotenanzahl
    - 62% der Anwendungen bei maximaler Knotenanzahl
  - gleiche Problematik bei allen anderen Sperren

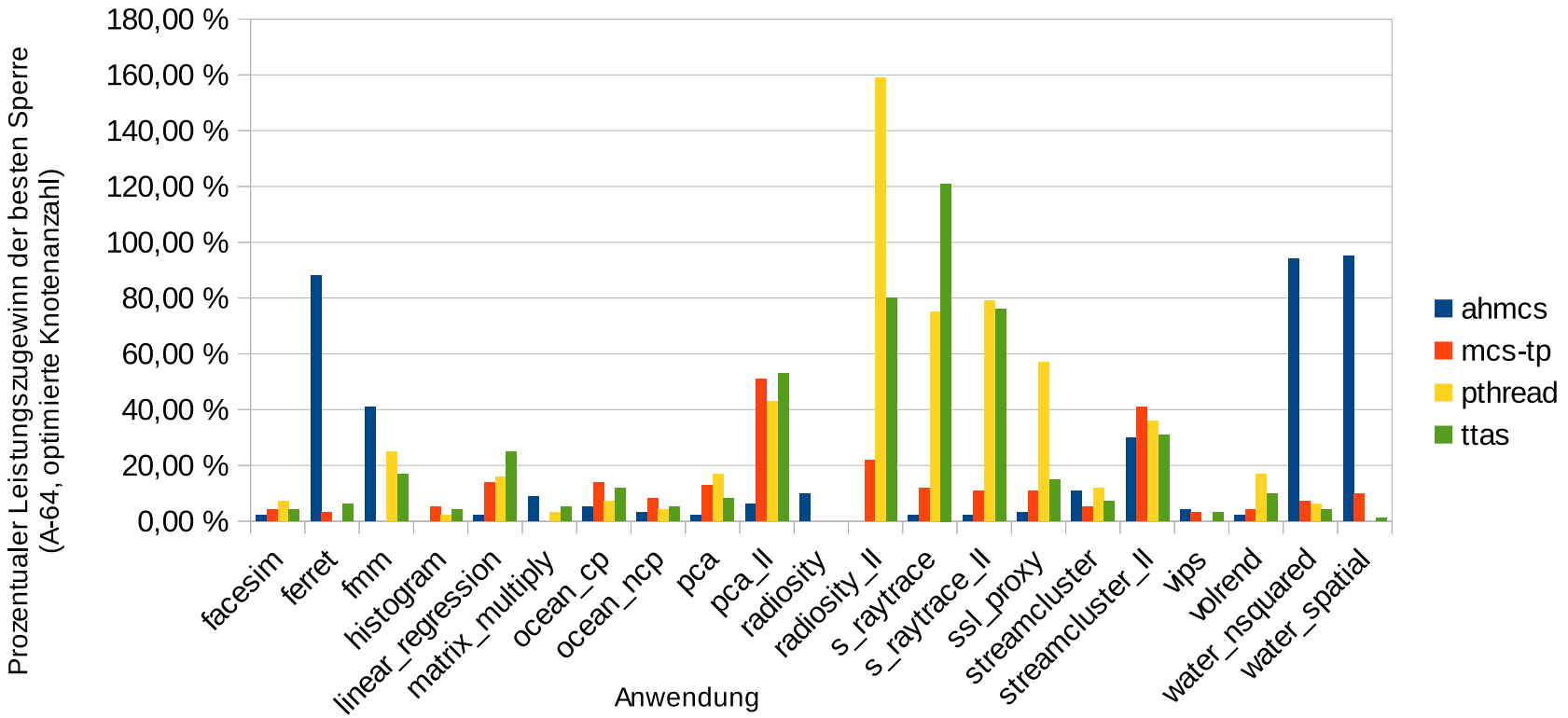
Beobachtung

Jede Sperre beeinflusst mehrere Anwendungen negativ

Ergebnisse

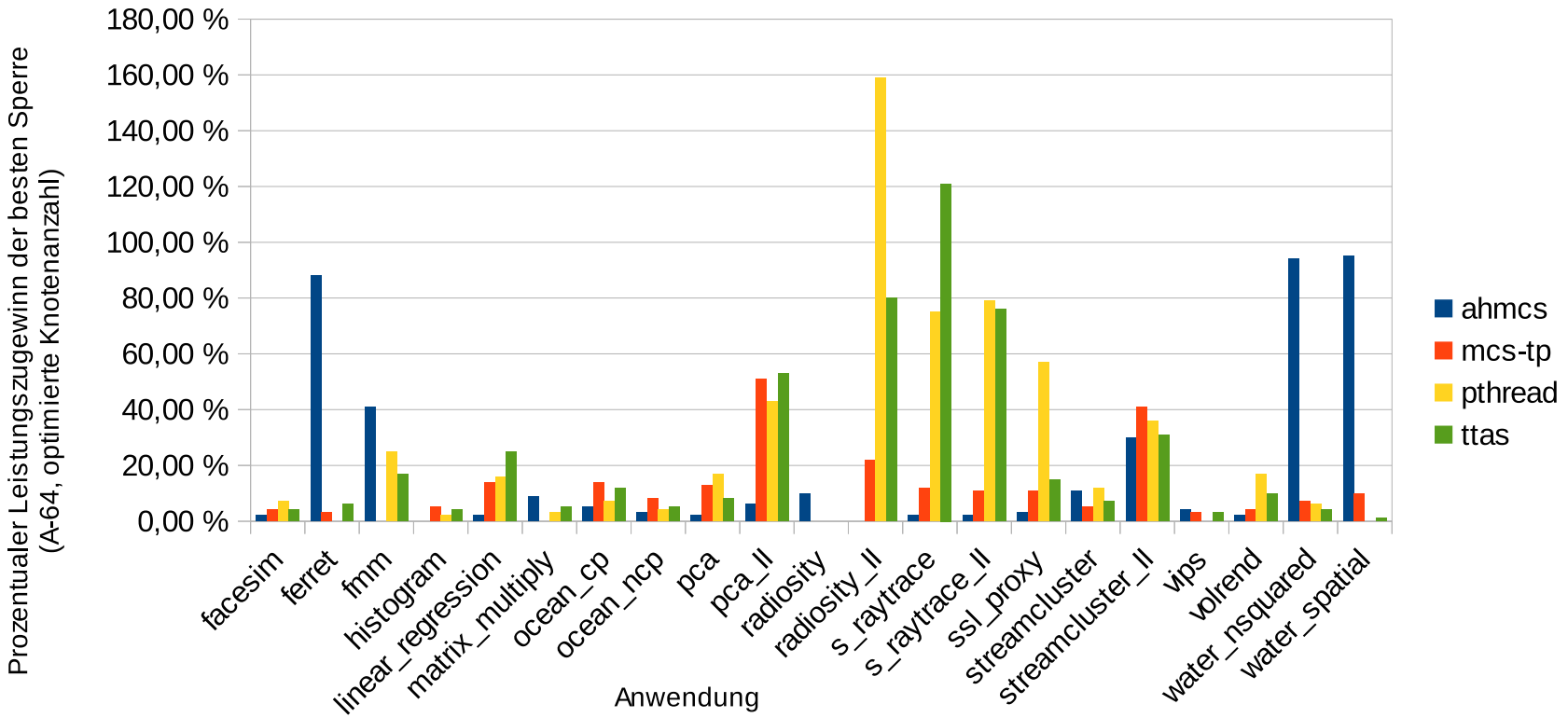

Prozentualer Leistungszugewinn der besten Sperre
(A-64, optimierte Knotenanzahl)
180,00 %
160,00 %
140,00 %
120,00 %
100,00 %
80,00 %
60,00 %
40,00 %
20,00 %
0,00 %
facesim
ferret
fmm
histogram
linear_regression
matrix_multiply
ocean_cp
ocean_ncp
pca
pca_ll
radiosity
radiosity_ll
s_raytrace
s_raytrace_ll
ssl_proxy
streamcluster
streamcluster_ll
vips
volrend
water_nsquared
water_spatial
Anwendung
ahmcs
mcs-tp
pthread
ttas

Ergebnisse



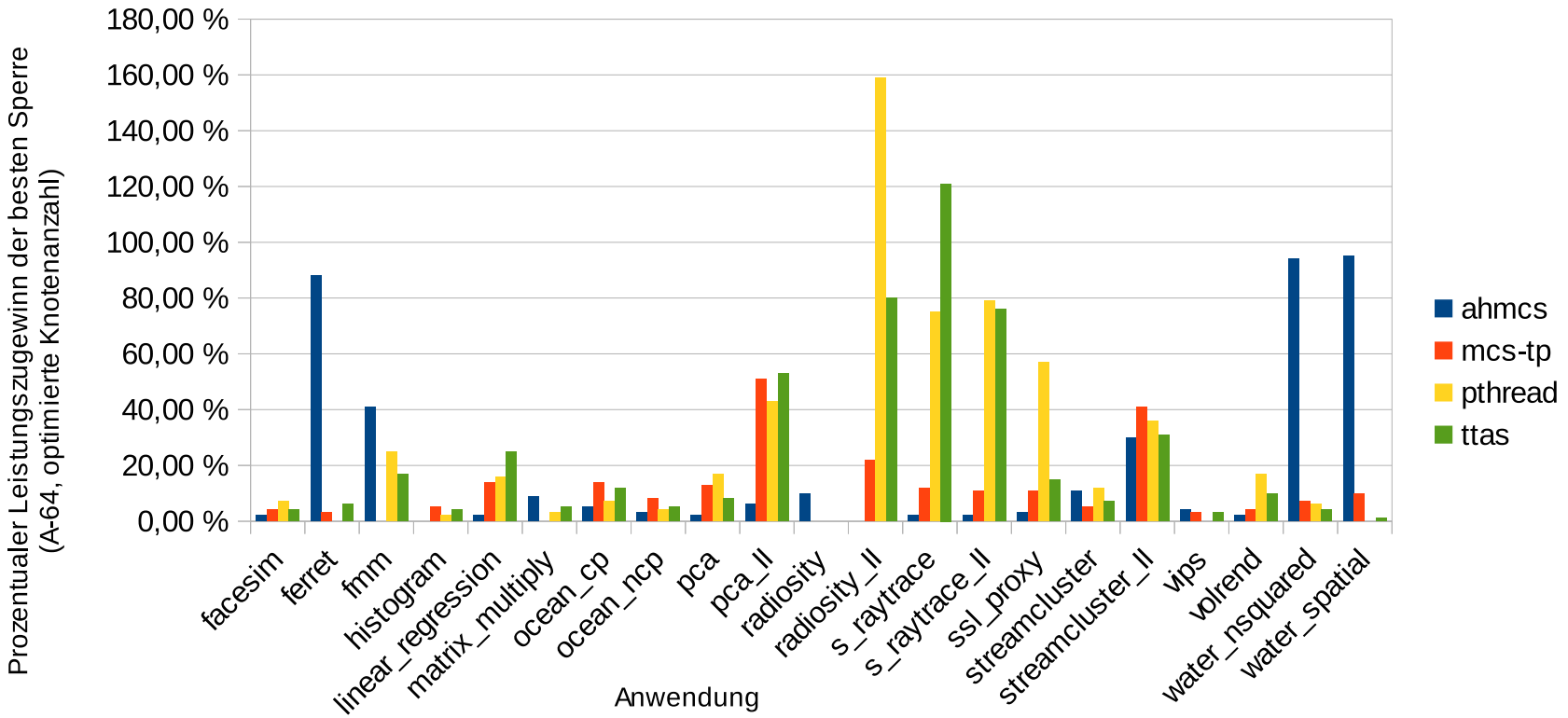
- Leistung von Pthread-Sperren

Ergebnisse



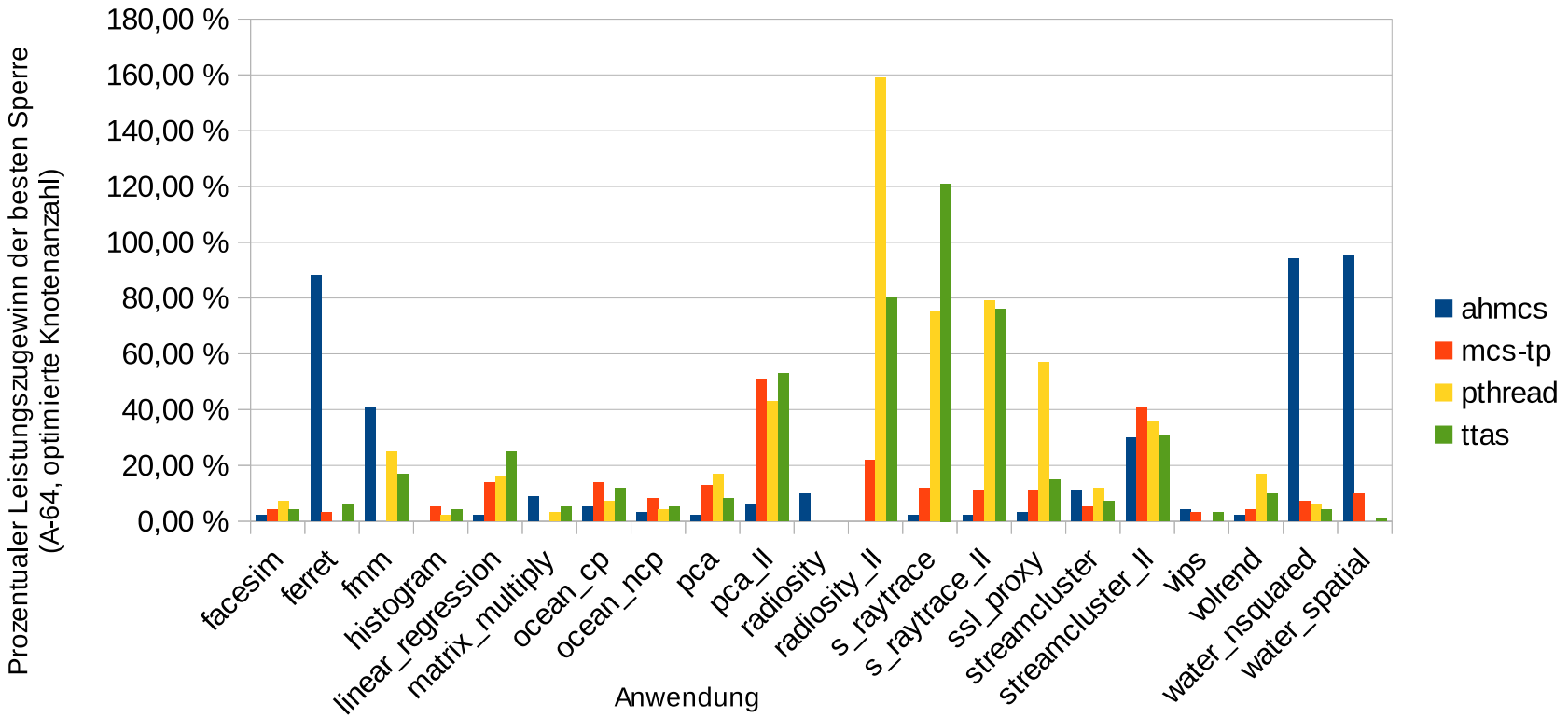
- Leistung von Pthread-Sperren
  - Pthread-Sperren bei den meisten Messungen nicht weit vom Durchschnitt

Ergebnisse



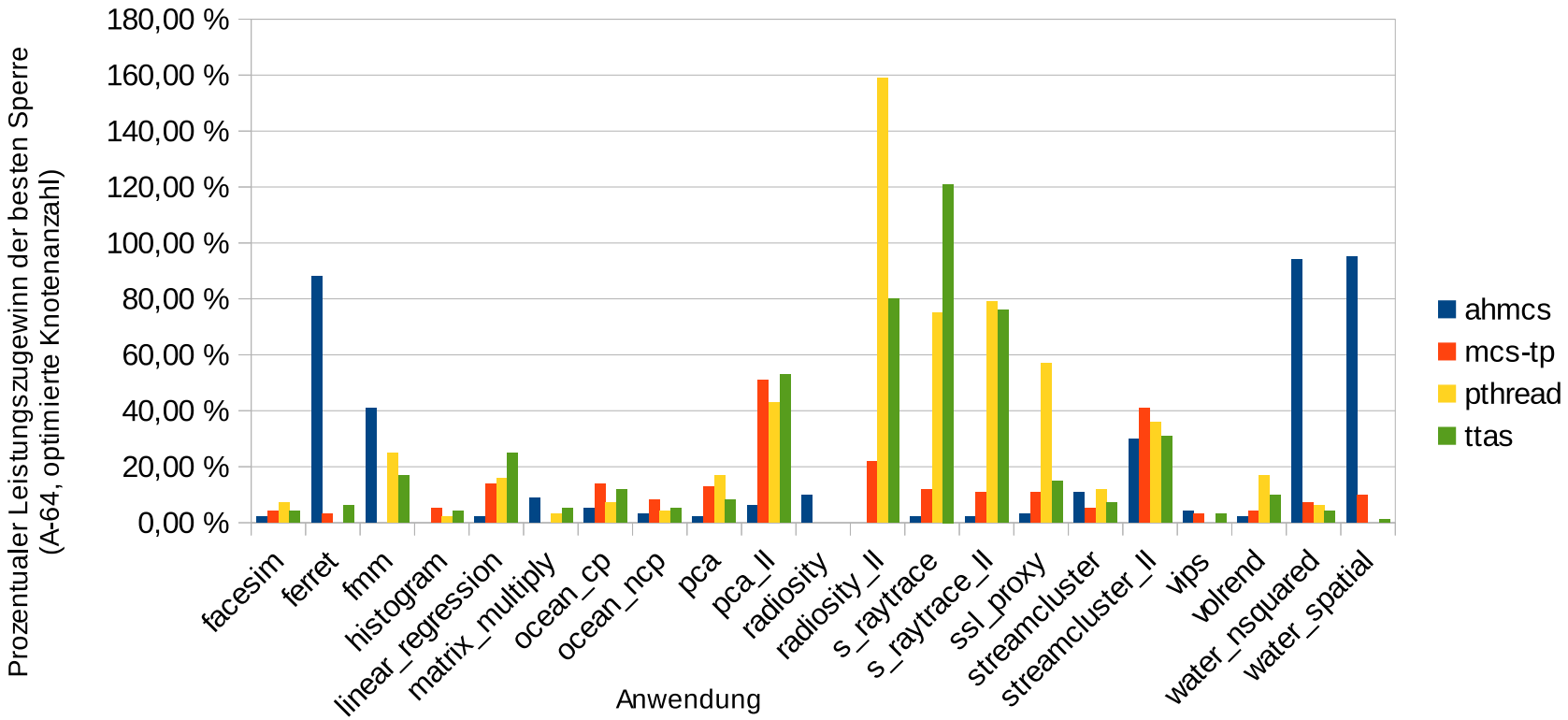
- Leistung von Pthread-Sperren
  - Pthread-Sperren bei den meisten Messungen nicht weit vom Durchschnitt
  - Pthread-Sperren unter den besten für einige Anwendungen

Ergebnisse



- Leistung von Pthread-Sperren
  - Pthread-Sperren bei den meisten Messungen nicht weit vom Durchschnitt
  - Pthread-Sperren unter den besten für einige Anwendungen
  - ähnliches Ergebnis auch bei Festlegung der Fäden auf einzelne Kerne

Ergebnisse



Beobachtung

Pthread-Sperren müssen nicht aufgrund schlechter Leistung vermieden werden

Fragen?