Evaluierung der Cache-Hierarchie eines nachrichtengekoppelten Manycore-Prozessors

Dominik Walter

27. September 2016

Inhaltsverzeichnis

Inhaltsverzeichnis

RCMC-Architektur

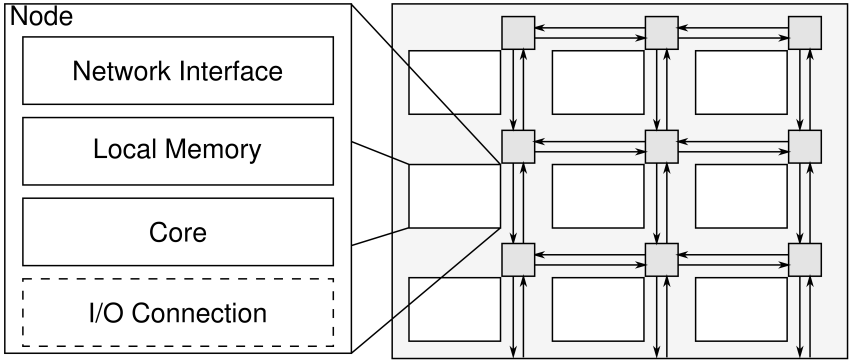


Abbildung: Aufbau der RCMC-Architektur

RCMC-Architektur

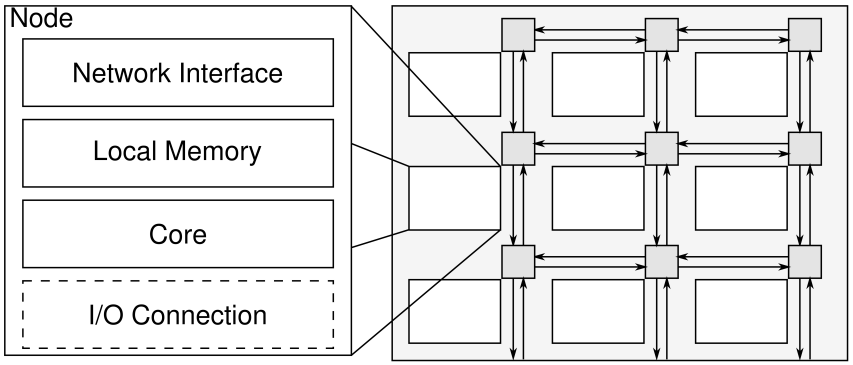


Abbildung: Aufbau der RCMC-Architektur

Eigenschaften

Original	Erweiterung
• lokaler Speicher	• lokaler Cache

RCMC-Architektur

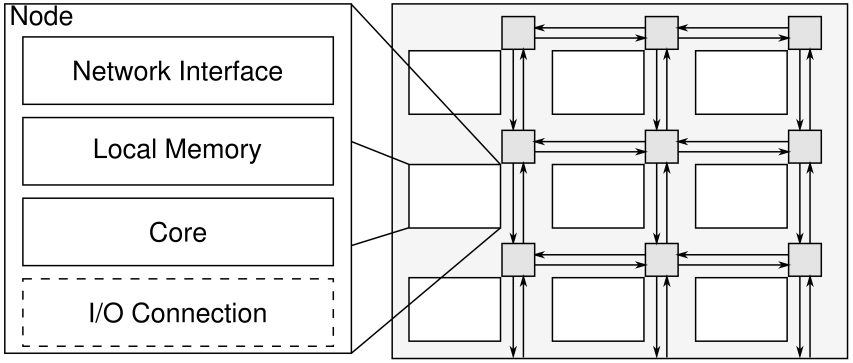


Abbildung: Aufbau der RCMC-Architektur

Eigenschaften

Original	Erweiterung
• lokaler Speicher	• lokaler Cache
• kein Hauptspeicher	• gemeinsamer Hauptspeicher

RCMC-Architektur

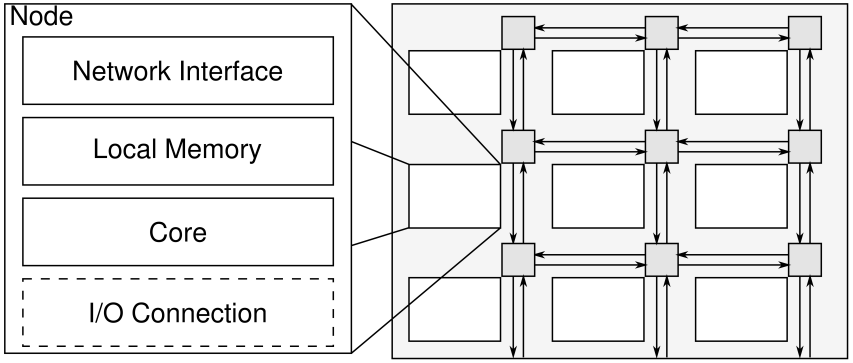


Abbildung: Aufbau der RCMC-Architektur

Eigenschaften

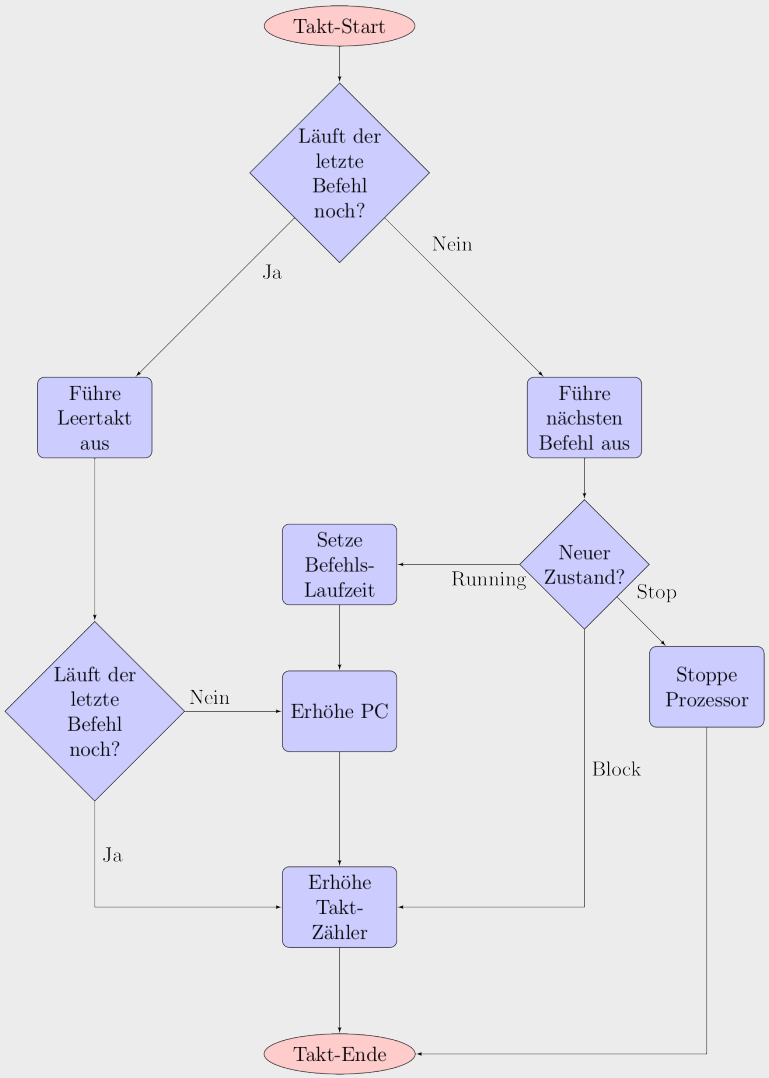
Original	Erweiterung
• lokaler Speicher	• lokaler Cache
• kein Hauptspeicher	• gemeinsamer Hauptspeicher
• Zugriffszeit immer 1 Takt	• realistische Zugriffszeit

Inhaltsverzeichnis

Inhaltsverzeichnis

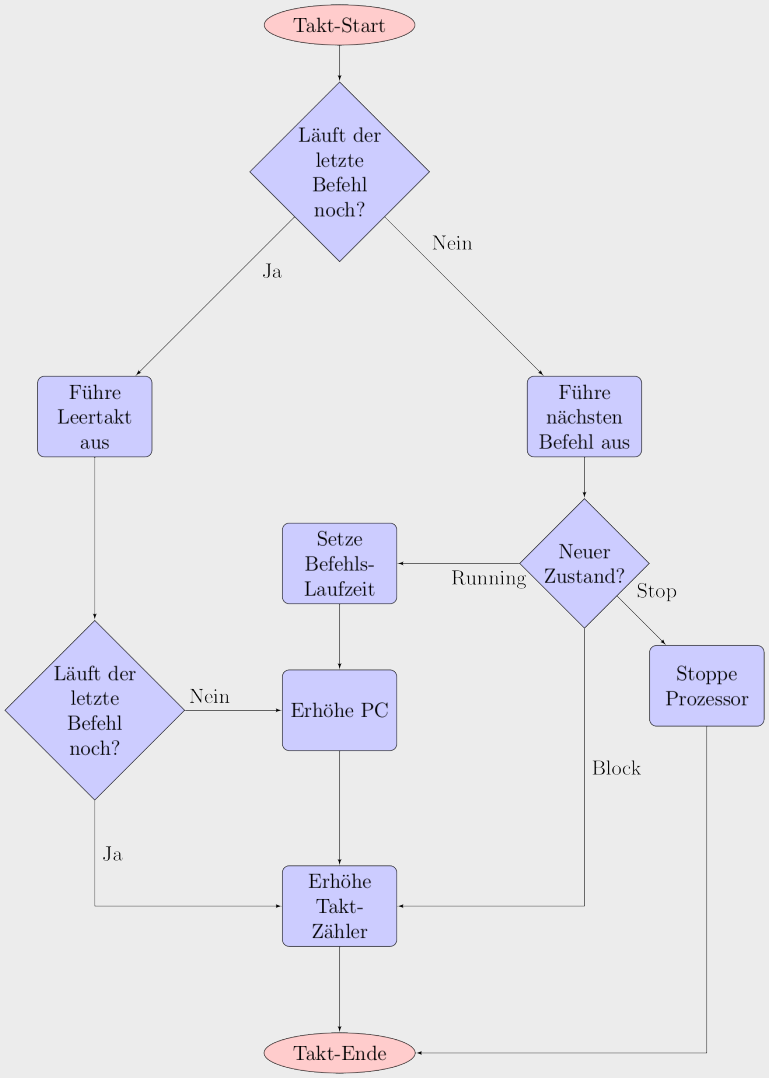
Simulator

Original

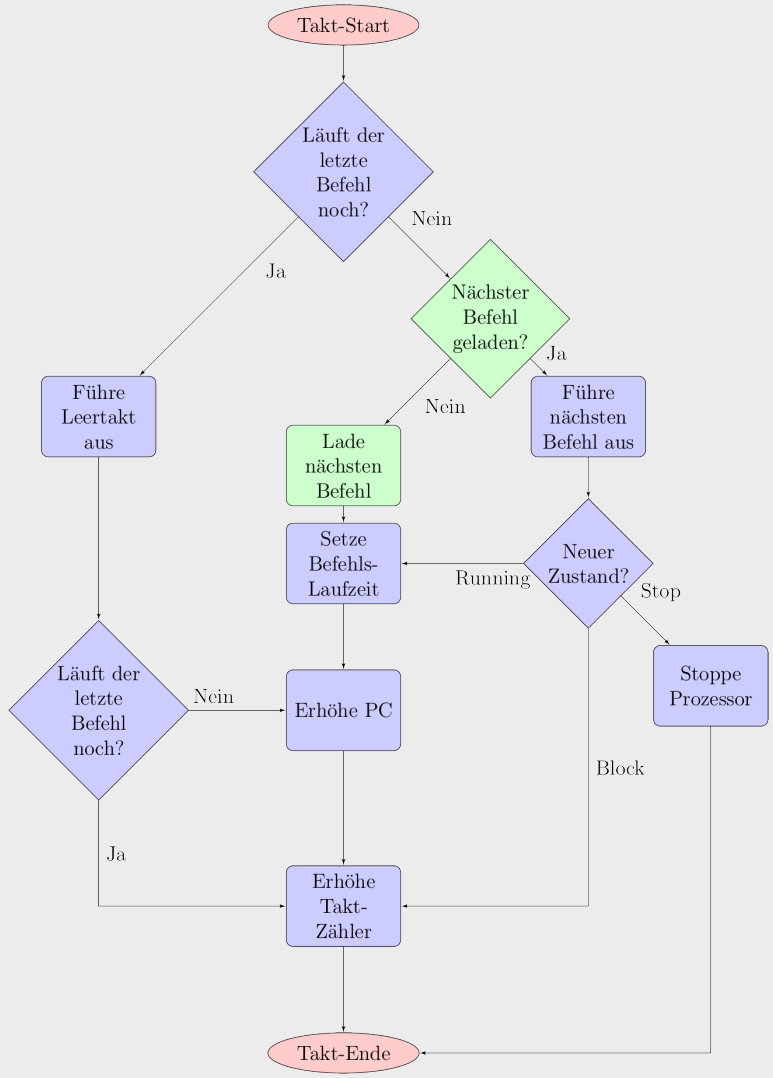


Simulator

Original



Mit Speichersystem



Inhaltsverzeichnis

Tests

Prozessor

- 4x4 Kerne

Tests

Prozessor

- 4x4 Kerne
- RCMC-Manycore

Tests

Prozessor

- 4x4 Kerne
- RCMC-Manycore

Hauptspeicher

- 1 Channel

Tests

Prozessor

- 4x4 Kerne
- RCMC-Manycore

Hauptspeicher

- 1 Channel
- 1 Queue

Tests

Prozessor

- 4x4 Kerne
- RCMC-Manycore

Hauptspeicher

- 1 Channel
- 1 Queue
- Hohe Zugriffszeit

Tests

Prozessor

- 4x4 Kerne
- RCMC-Manycore

Hauptspeicher

- 1 Channel
- 1 Queue
- Hohe Zugriffszeit

Benchmarks

- Integer Sort (IS)

Tests

Prozessor

- 4x4 Kerne
- RCMC-Manycore

Hauptspeicher

- 1 Channel
- 1 Queue
- Hohe Zugriffszeit

Benchmarks

- Integer Sort (IS)
- Data Traffic (DT)

Tests

Prozessor

- 4x4 Kerne
- RCMC-Manycore

Hauptspeicher

- 1 Channel
- 1 Queue
- Hohe Zugriffszeit

Benchmarks

- Integer Sort (IS)
- Data Traffic (DT)
- Conjugate Gradient (CG)

Inhaltsverzeichnis

Schreibstrategie

Problematik

- Wann werden die Einträge im Hauptspeicher aktualisiert?

Schreibstrategie

Problematik

- Wann werden die Einträge im Hauptspeicher aktualisiert?

Lösung I: Write-Through

- Daten werden immer direkt in den RAM schreiben

Schreibstrategie

Problematik

- Wann werden die Einträge im Hauptspeicher aktualisiert?

Lösung I: Write-Through

- Daten werden immer direkt in den RAM schreiben
- Im Speicher steht immer der aktuelle Wert

Schreibstrategie

Problematik

- Wann werden die Einträge im Hauptspeicher aktualisiert?

Lösung I: Write-Through

- Daten werden immer direkt in den RAM schreiben
- Im Speicher steht immer der aktuelle Wert

Lösung II: Write-Back

- Erst bei einer Verdrängung wird zurückgeschrieben

Schreibstrategie

Problematik

- Wann werden die Einträge im Hauptspeicher aktualisiert?

Lösung I: Write-Through

- Daten werden immer direkt in den RAM schreiben
- Im Speicher steht immer der aktuelle Wert

Lösung II: Write-Back

- Erst bei einer Verdrängung wird zurückgeschrieben
- Weniger Traffic

Schreibstrategie

Problematik

- Wann werden die Einträge im Hauptspeicher aktualisiert?

Lösung I: Write-Through

- Daten werden immer direkt in den RAM schreiben
- Im Speicher steht immer der aktuelle Wert

Lösung II: Write-Back

- Erst bei einer Verdrängung wird zurückgeschrieben
- Weniger Traffic
- Aufwändiger zu Implementieren

Schreibstrategie

Problematik

- Wird beim Schreiben der Eintrag in den Cache geladen?

Schreibstrategie

Problematik

- Wird beim Schreiben der Eintrag in den Cache geladen?

Lösung I: Write-Allocate

- Bei einem Schreibzugriff wird der Eintrag in den Cache geladen

Schreibstrategie

Problematik

- Wird beim Schreiben der Eintrag in den Cache geladen?

Lösung I: Write-Allocate

- Bei einem Schreibzugriff wird der Eintrag in den Cache geladen
- Nur mit Write-Back sinnvoll

Schreibstrategie

Problematik

- Wird beim Schreiben der Eintrag in den Cache geladen?

Lösung I: Write-Allocate

- Bei einem Schreibzugriff wird der Eintrag in den Cache geladen
- Nur mit Write-Back sinnvoll
- Folgen auf einen Schreibzugriff weitere Zugriffe, so befindet sich der Eintrag bereits im Cache

Schreibstrategie

Problematik

- Wird beim Schreiben der Eintrag in den Cache geladen?

Lösung I: Write-Allocate

- Bei einem Schreibzugriff wird der Eintrag in den Cache geladen
- Nur mit Write-Back sinnvoll
- Folgen auf einen Schreibzugriff weitere Zugriffe, so befindet sich der Eintrag bereits im Cache

Lösung II: Write-Non-Allocate

- Bei einem Schreibzugriff wird der Eintrag nicht in den Cache geladen

Schreibstrategie

Problematik

- Wird beim Schreiben der Eintrag in den Cache geladen?

Lösung I: Write-Allocate

- Bei einem Schreibzugriff wird der Eintrag in den Cache geladen
- Nur mit Write-Back sinnvoll
- Folgen auf einen Schreibzugriff weitere Zugriffe, so befindet sich der Eintrag bereits im Cache

Lösung II: Write-Non-Allocate

- Bei einem Schreibzugriff wird der Eintrag nicht in den Cache geladen
- Eventuell unnötiger Eintrag wird nicht in den Cache geladen

Schreibstrategie

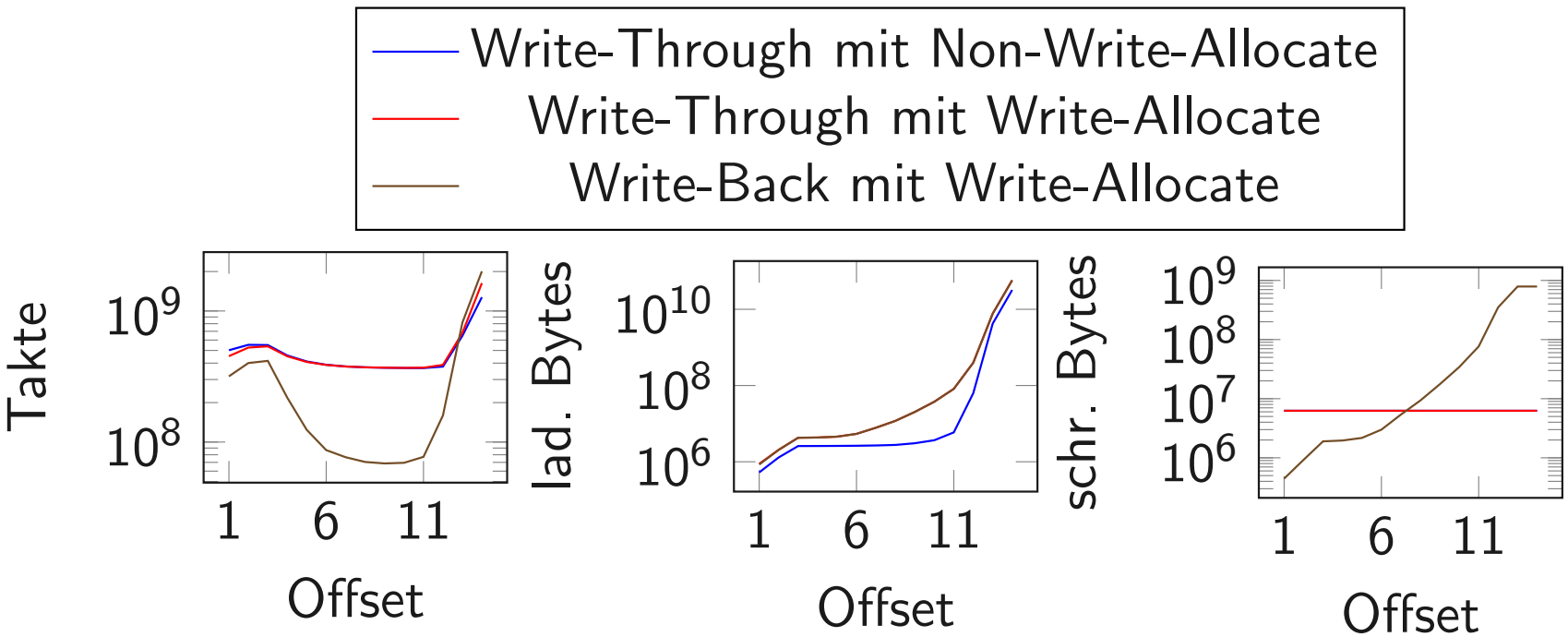


Abbildung: Vergleich zwischen verschiedenen Schreibstrategien

Inhaltsverzeichnis

Cache-Line Größe

Problematik

- Wie groß sollte eine Cache-Line sein?

Cache-Line Größe

Problematik

- Wie groß sollte eine Cache-Line sein?

Lösung I: Kleine Cache-Line

- Bei einer Verdrängung werden weniger Daten ersetzt

Cache-Line Größe

Problematik

- Wie groß sollte eine Cache-Line sein?

Lösung I: Kleine Cache-Line

- Bei einer Verdrängung werden weniger Daten ersetzt
- Schlechte Ausnutzung der Lokalitäts-Eigenschaft

Cache-Line Größe

Problematik

- Wie groß sollte eine Cache-Line sein?

Lösung I: Kleine Cache-Line

- Bei einer Verdrängung werden weniger Daten ersetzt
- Schlechte Ausnutzung der Lokalitäts-Eigenschaft

Lösung II: Große Cache-Line

- Bessere Ausnutzung der Lokalitäts-Eigenschaft

Cache-Line Größe

Problematik

- Wie groß sollte eine Cache-Line sein?

Lösung I: Kleine Cache-Line

- Bei einer Verdrängung werden weniger Daten ersetzt
- Schlechte Ausnutzung der Lokalitäts-Eigenschaft

Lösung II: Große Cache-Line

- Bessere Ausnutzung der Lokalitäts-Eigenschaft
- Bei Cache-Misses müssen mehr Daten geladen werden

Cache-Line Größe

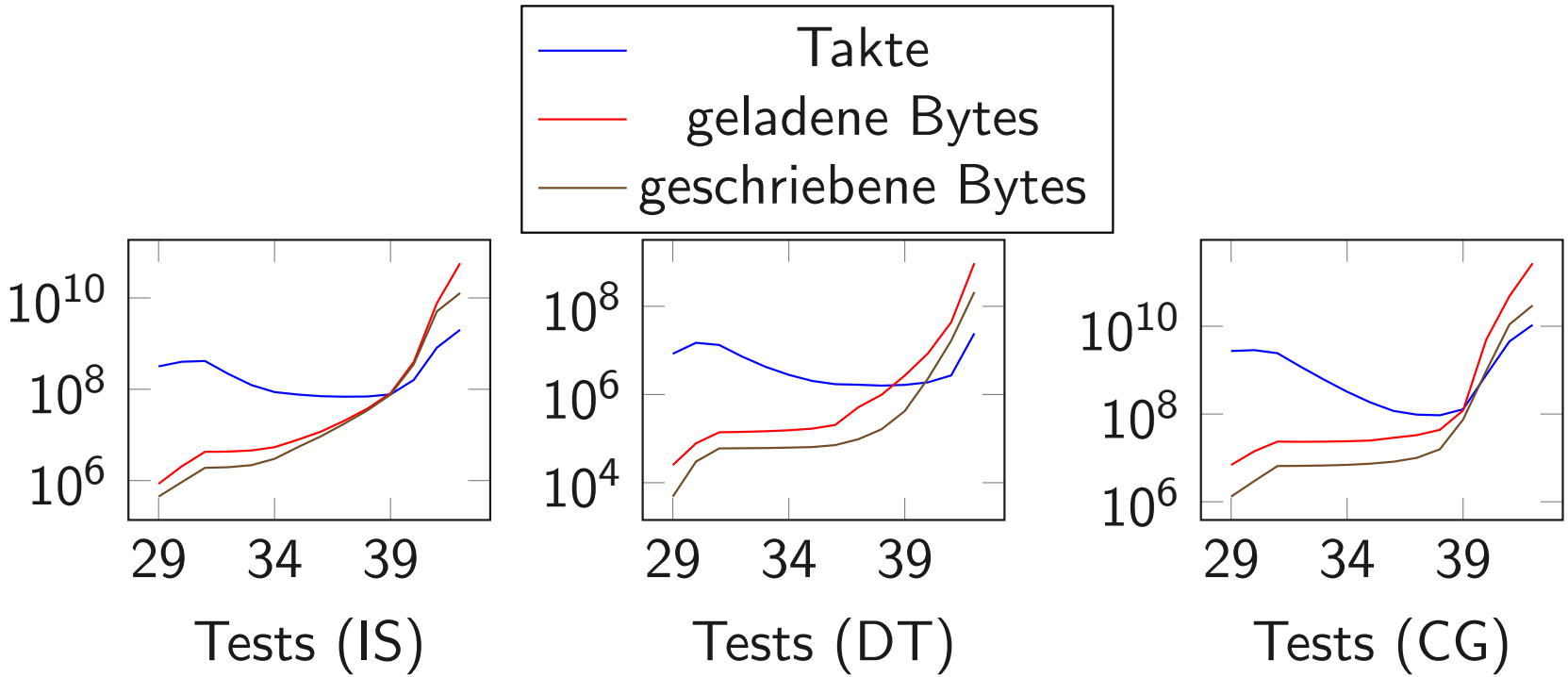


Abbildung: Für größere *Cache-Lines* steigt der *Traffic*, während die Takte bei Test 38 minimal sind.

Inhaltsverzeichnis

Assoziativität

Problematik

- Wieviele Einträge müssen bei einem Zugriff verglichen werden?

Assoziativität

Problematik

- Wieviele Einträge müssen bei einem Zugriff verglichen werden?

Lösung I: Direkt-Abgebildet

- 1 Eintrag

Assoziativität

Problematik

- Wieviele Einträge müssen bei einem Zugriff verglichen werden?

Lösung I: Direkt-Abgebildet

- 1 Eintrag
- Schlechte Ausnutzung des Cache-Speichers

Assoziativität

Problematik

- Wieviele Einträge müssen bei einem Zugriff verglichen werden?

Lösung I: Direkt-Abgebildet

- 1 Eintrag
- Schlechte Ausnutzung des Cache-Speichers

Lösung II: n-Fach Satz-Assoziativ

- n Einträge

Assoziativität

Problematik

- Wieviele Einträge müssen bei einem Zugriff verglichen werden?

Lösung I: Direkt-Abgebildet

- 1 Eintrag
- Schlechte Ausnutzung des Cache-Speichers

Lösung II: n-Fach Satz-Assoziativ

- n Einträge
- Bessere Ausnutzung des Cache-Speichers

Assoziativität

Problematik

- Wieviele Einträge müssen bei einem Zugriff verglichen werden?

Lösung I: Direkt-Abgebildet

- 1 Eintrag
- Schlechte Ausnutzung des Cache-Speichers

Lösung II: n-Fach Satz-Assoziativ

- n Einträge
- Bessere Ausnutzung des Cache-Speichers

Lösung III: Voll-Assoziativ

- alle Einträge

Assoziativität

Problematik

- Wieviele Einträge müssen bei einem Zugriff verglichen werden?

Lösung I: Direkt-Abgebildet

- 1 Eintrag
- Schlechte Ausnutzung des Cache-Speichers

Lösung II: n-Fach Satz-Assoziativ

- n Einträge
- Bessere Ausnutzung des Cache-Speichers

Lösung III: Voll-Assoziativ

- alle Einträge
- Beste Ausnutzung des Cache-Speichers

Assoziativität

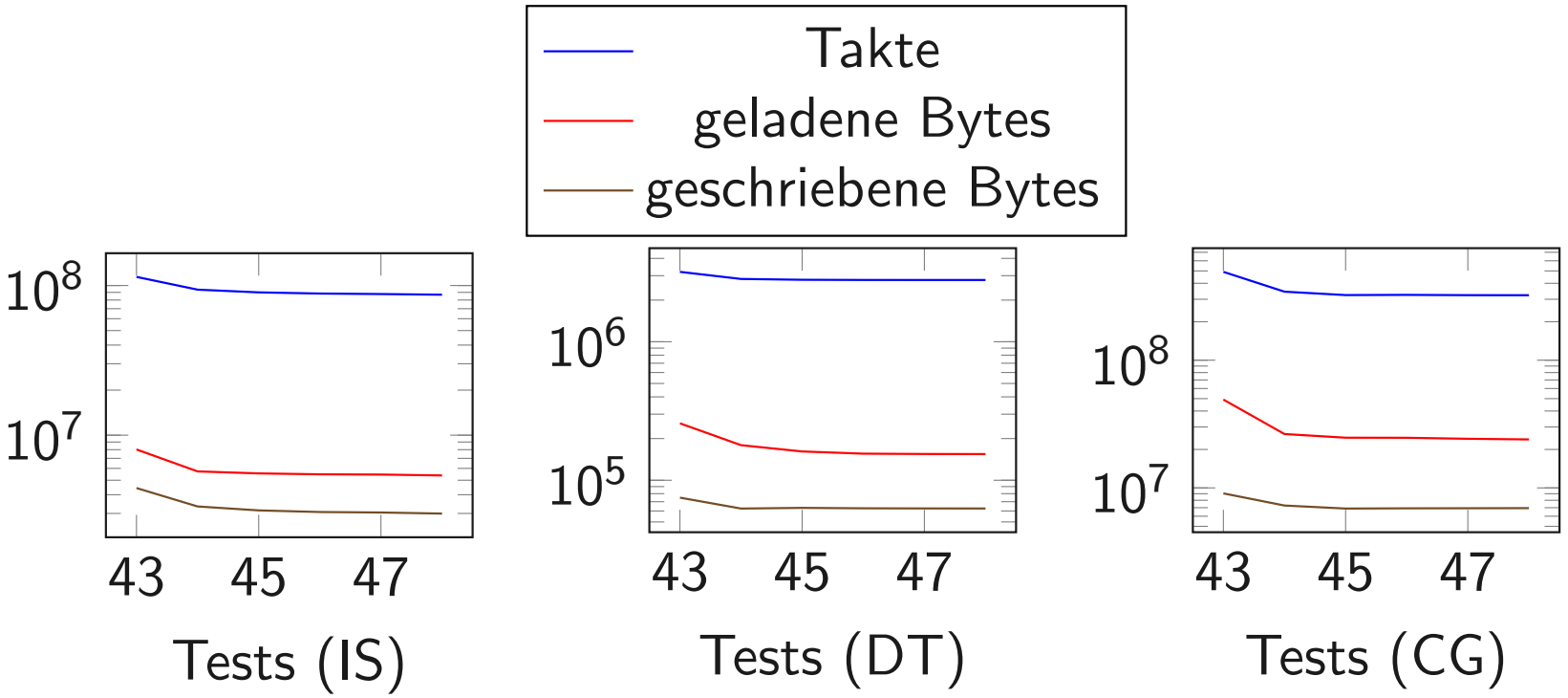


Abbildung: Die Satzgröße hat kaum Einfluss auf die *Benchmarks*

Inhaltsverzeichnis

Cache Größe

Problematik

- Wie groß muss der Cache mindestens sein?

Cache Größe

Problematik

- Wie groß muss der Cache mindestens sein?
- Gibt es eine maximale Größe?

Cache Größe

Problematik

- Wie groß muss der Cache mindestens sein?
- Gibt es eine maximale Größe?

Lösung

- So groß wie möglich

Cache Größe

Problematik

- Wie groß muss der Cache mindestens sein?
- Gibt es eine maximale Größe?

Lösung

- So groß wie möglich
- Platz auf Chip ist begrenzt

Cache Größe

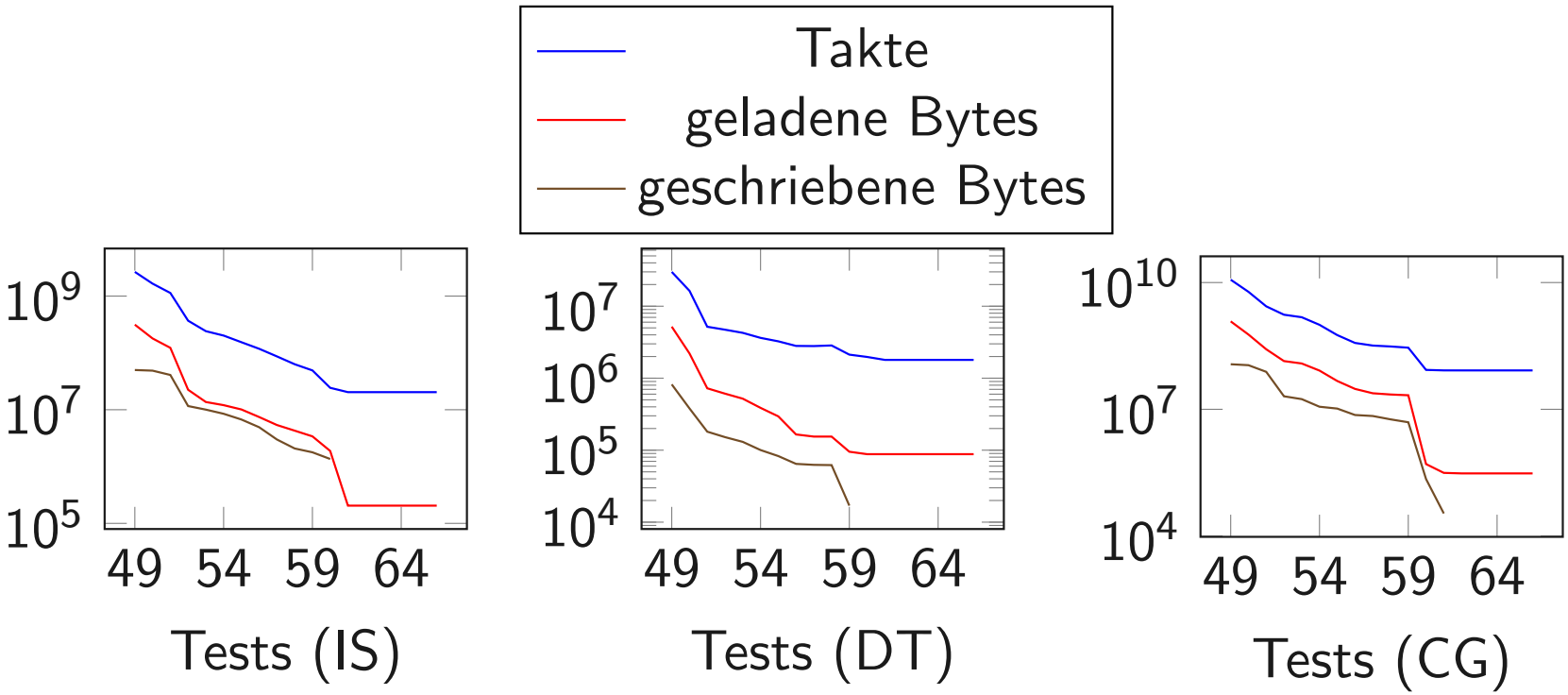


Abbildung: Stetige Verbesserung durch Vergrößerung des Caches

Inhaltsverzeichnis

Zugriffszeit

Problematik

- Wann schnell muss der Cache sein um sich noch zu lohnen?

Zugriffszeit

Problematik

- Wann schnell muss der Cache sein um sich noch zu lohnen?
- Ist ein kleiner schneller Cache besser als ein Größerer?

Zugriffszeit

Problematik

- Wann schnell muss der Cache sein um sich noch zu lohnen?
- Ist ein kleiner schneller Cache besser als ein Größerer?

Lösung I: Klein und Schnell

- Kürzere Zugriffszeit

Zugriffszeit

Problematik

- Wann schnell muss der Cache sein um sich noch zu lohnen?
- Ist ein kleiner schneller Cache besser als ein Größerer?

Lösung I: Klein und Schnell

- Kürzere Zugriffszeit
- Niedrigere Hit-Rate

Zugriffszeit

Problematik

- Wann schnell muss der Cache sein um sich noch zu lohnen?
- Ist ein kleiner schneller Cache besser als ein Größerer?

Lösung I: Klein und Schnell

- Kürzere Zugriffszeit
- Niedrigere Hit-Rate

Lösung II: Groß und Langsam

- Längere Zugriffszeit

Zugriffszeit

Problematik

- Wann schnell muss der Cache sein um sich noch zu lohnen?
- Ist ein kleiner schneller Cache besser als ein Größerer?

Lösung I: Klein und Schnell

- Kürzere Zugriffszeit
- Niedrigere Hit-Rate

Lösung II: Groß und Langsam

- Längere Zugriffszeit
- Höhere Hit-Rate

Zugriffszeit

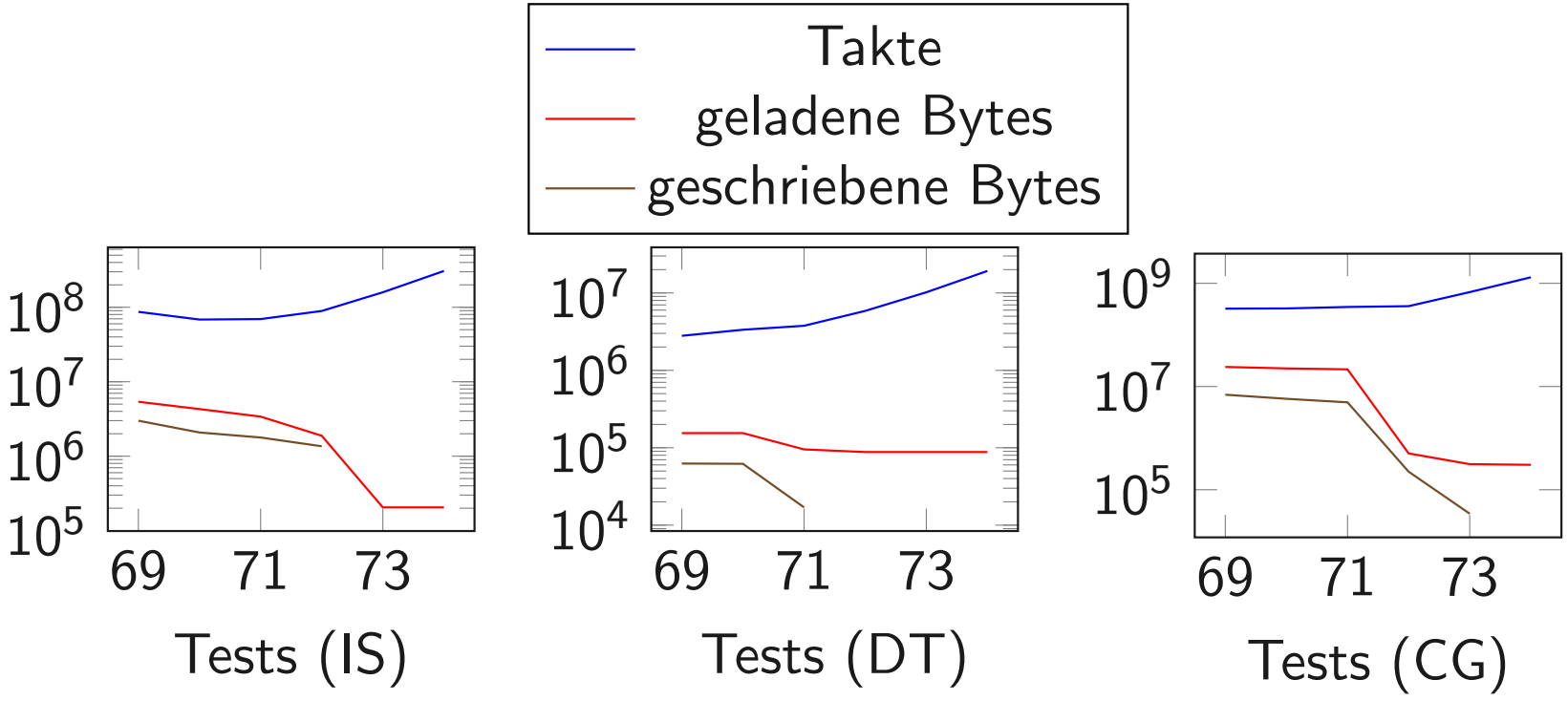


Abbildung: Geschwindigkeit ist für die Ausführungszeit wichtiger als die Größe

Inhaltsverzeichnis

Cache-Hierarchie

Problematik

- Wieviele Ebenen sind sinnvoll?

Cache-Hierarchie

Problematik

- Wieviele Ebenen sind sinnvoll?
- Welche Strategie lohnt mehr?

Cache-Hierarchie

Problematik

- Wieviele Ebenen sind sinnvoll?
- Welche Strategie lohnt mehr?

Lösung I: Exklusiv

- Keine Duplikate

Cache-Hierarchie

Problematik

- Wieviele Ebenen sind sinnvoll?
- Welche Strategie lohnt mehr?

Lösung I: Exklusiv

- Keine Duplikate
- Höhere Gesamtgröße

Cache-Hierarchie

Problematik

- Wieviele Ebenen sind sinnvoll?
- Welche Strategie lohnt mehr?

Lösung I: Exklusiv

- Keine Duplikate
- Höhere Gesamtgröße
- Jeder verdrängte Eintrag muss in die nächste Ebene geschrieben werden

Cache-Hierarchie

Problematik

- Wieviele Ebenen sind sinnvoll?
- Welche Strategie lohnt mehr?

Lösung I: Exklusiv

- Keine Duplikate
- Höhere Gesamtgröße
- Jeder verdrängte Eintrag muss in die nächste Ebene geschrieben werden

Lösung II: Inklusiv

- Jede Ebene beinhaltet alle Einträge des Vorgängers

Cache-Hierarchie

Problematik

- Wieviele Ebenen sind sinnvoll?
- Welche Strategie lohnt mehr?

Lösung I: Exklusiv

- Keine Duplikate
- Höhere Gesamtgröße
- Jeder verdrängte Eintrag muss in die nächste Ebene geschrieben werden

Lösung II: Inklusiv

- Jede Ebene beinhaltet alle Einträge des Vorgängers
- Kleinere Gesamtgröße

Cache-Hierarchie

Problematik

- Wieviele Ebenen sind sinnvoll?
- Welche Strategie lohnt mehr?

Lösung I: Exklusiv

- Keine Duplikate
- Höhere Gesamtgröße
- Jeder verdrängte Eintrag muss in die nächste Ebene geschrieben werden

Lösung II: Inklusiv

- Jede Ebene beinhaltet alle Einträge des Vorgängers
- Kleinere Gesamtgröße
- Nur veränderte Einträge werden bei einer Verdrängung in die nächste Ebene geschrieben

Cache-Hierarchie

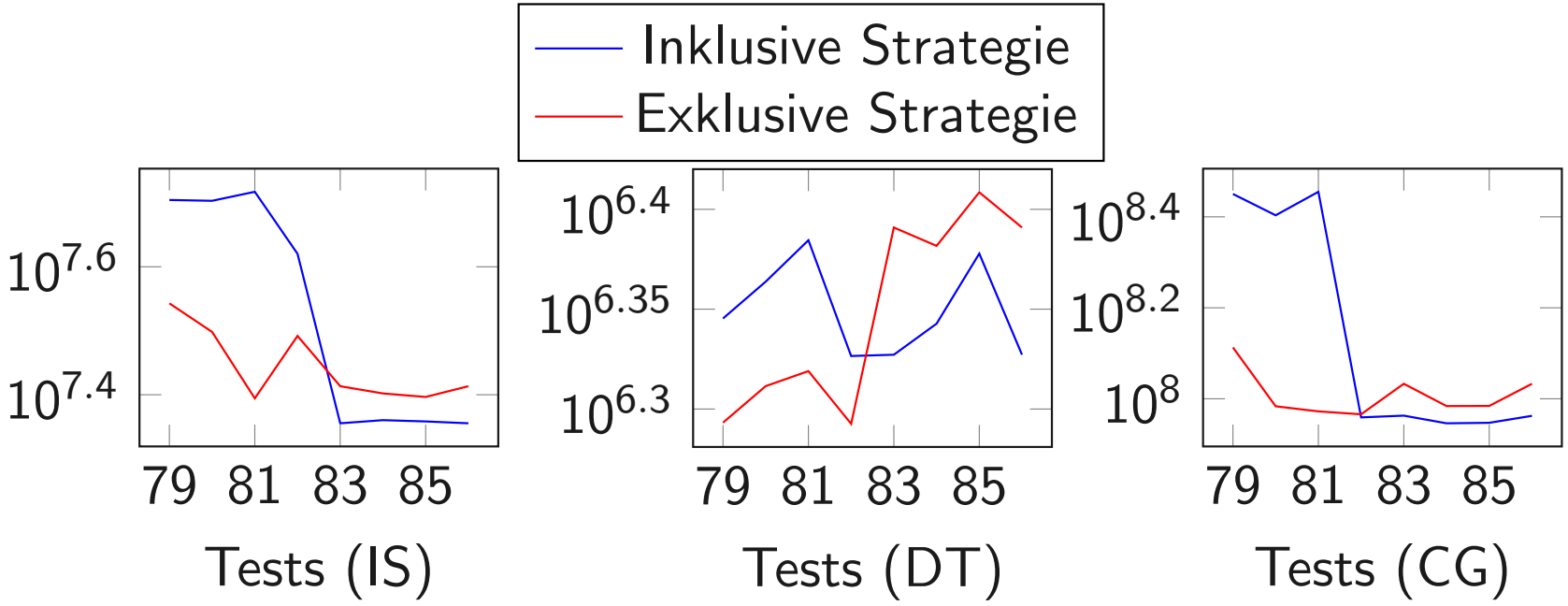


Abbildung: Erst für große *Caches* (Test 83 - 86) lohnt sich eine inklusive Strategie

Inhaltsverzeichnis

Auswertung

Beispiel: optimaler Cache

- Schreibstrategie: Write-Back mit Write-Allocate

Auswertung

Beispiel: optimaler Cache

- Schreibstrategie: Write-Back mit Write-Allocate
- Cache-Line Größe: 64 Byte

Auswertung

Beispiel: optimaler Cache

- Schreibstrategie: Write-Back mit Write-Allocate
- Cache-Line Größe: 64 Byte
- Assoziativität: 16x Satz-Assoziativ

Auswertung

Beispiel: optimaler Cache

• Schreibstrategie:	Write-Back mit Write-Allocate
• Cache-Line Größe:	64 Byte
• Assoziativität:	16x Satz-Assoziativ
• Cache Größe:	L1: 4 KByte L2: 8 KByte

Auswertung

Beispiel: optimaler Cache

• Schreibstrategie:	Write-Back mit Write-Allocate
• Cache-Line Größe:	64 Byte
• Assoziativität:	16x Satz-Assoziativ
• Cache Größe:	L1: 4 KByte L2: 8 KByte
• Zugriffszeit:	Niedrig

Auswertung

Beispiel: optimaler Cache

• Schreibstrategie:	Write-Back mit Write-Allocate
• Cache-Line Größe:	64 Byte
• Assoziativität:	16x Satz-Assoziativ
• Cache Größe:	L1: 4 KByte L2: 8 KByte
• Zugriffszeit:	Niedrig
• Cache-Hierarchie:	Exklusiv

Auswertung

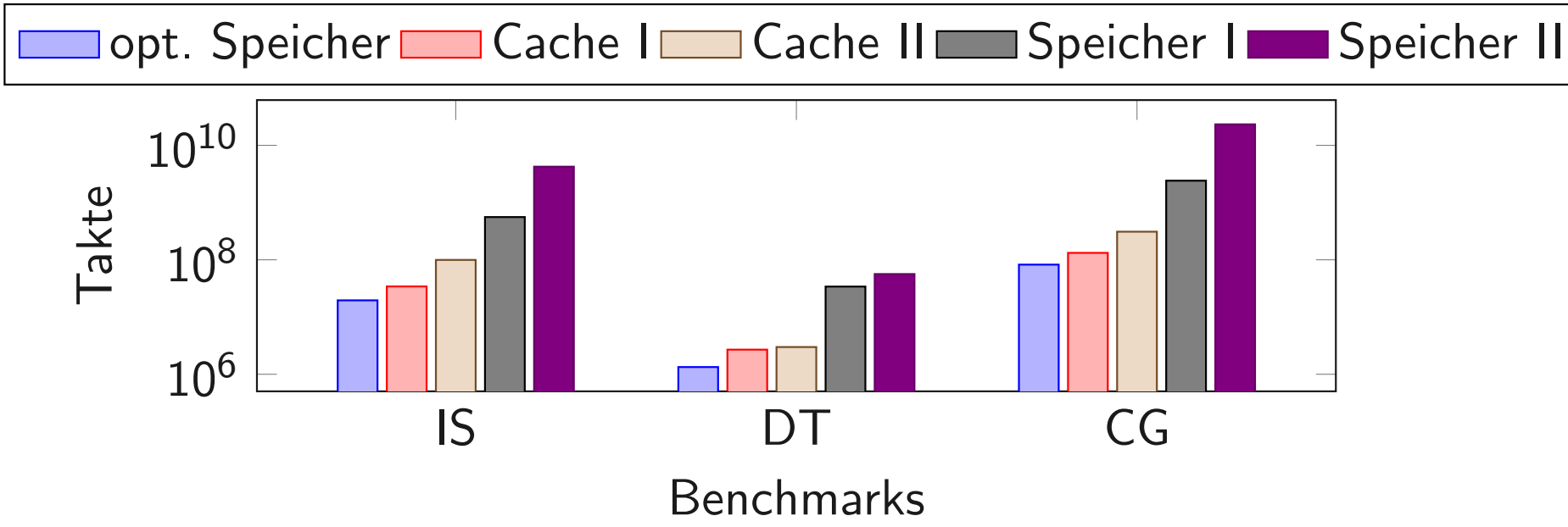


Abbildung: Ausführungszeit mit unterschiedlichen Speichersystemen.

Auswertung

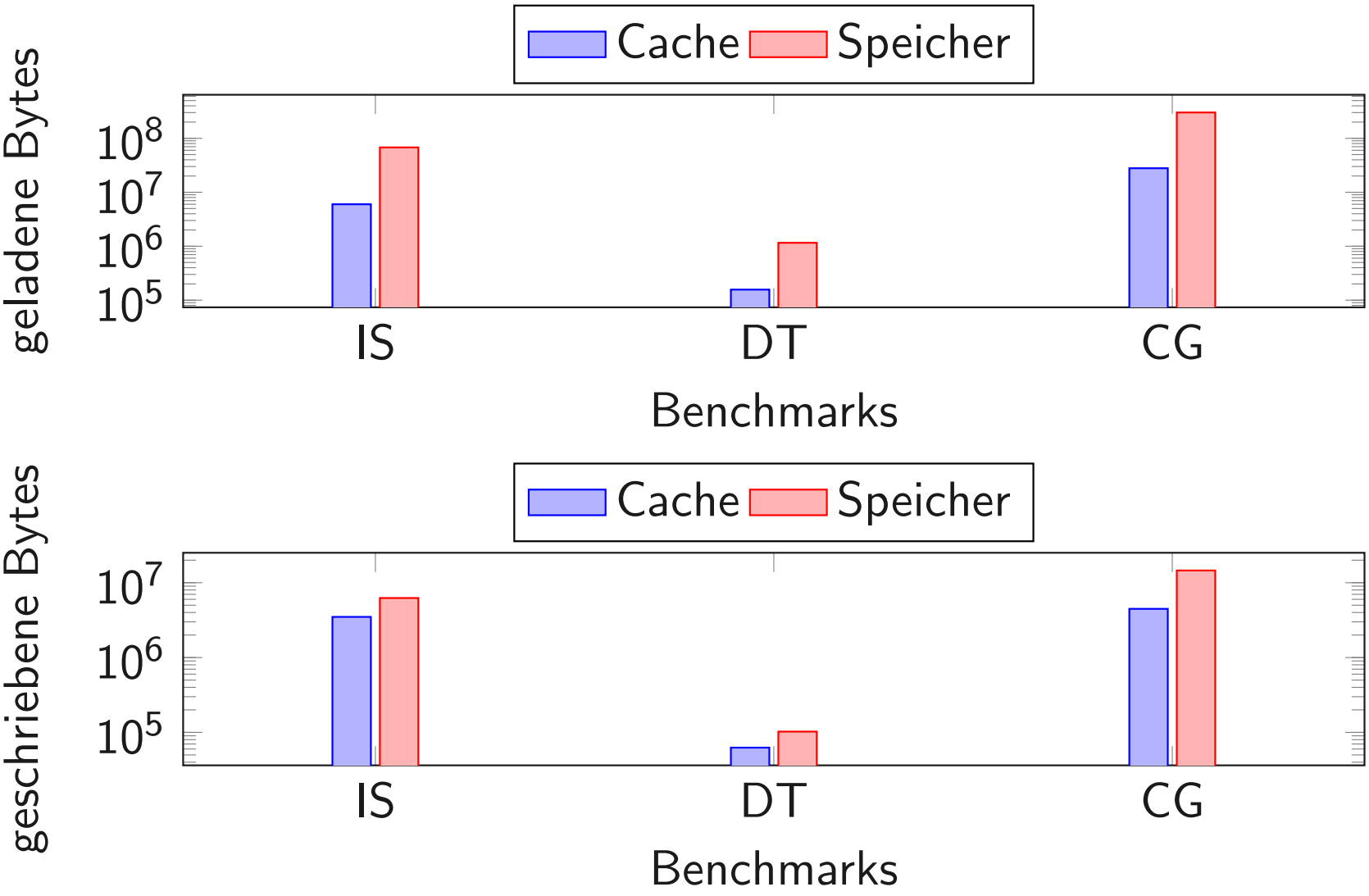


Abbildung: Traffic mit unterschiedlichen Speichersystemen.

Auswertung

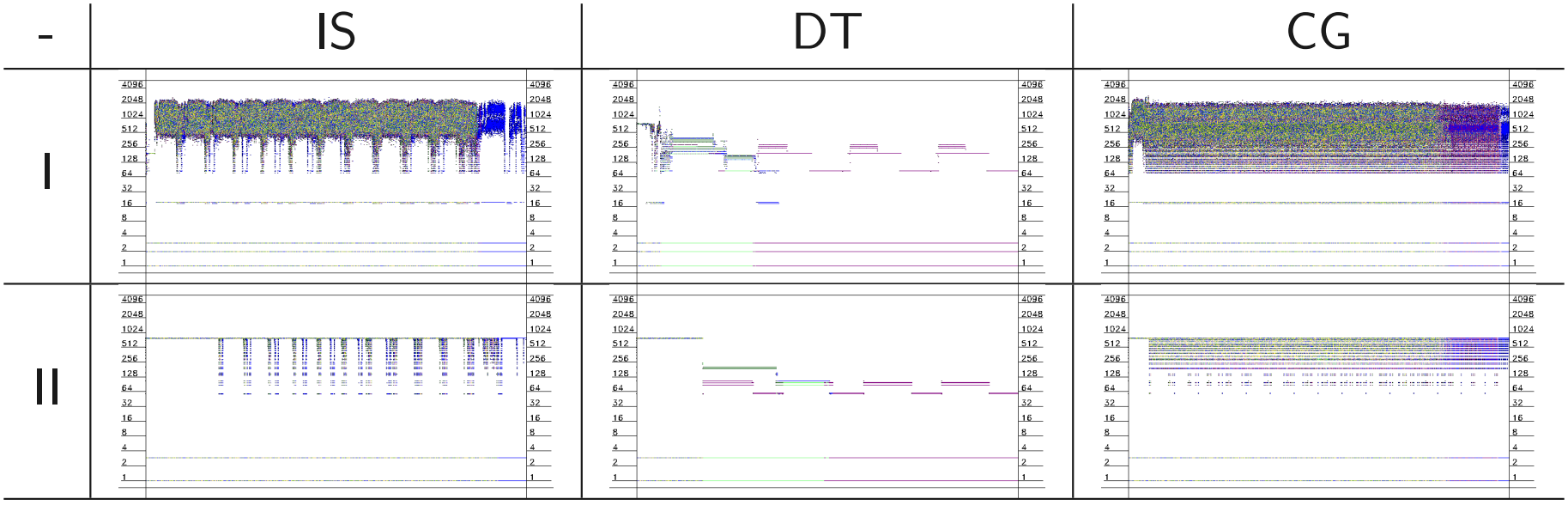


Abbildung: Programmablauf auf verschiedenen Systemen (I: mit Cache — II: ohne Cache)

Inhaltsverzeichnis

Fazit

Vorteile

- Kürzere Ausführungszeit

Fazit

Vorteile

- Kürzere Ausführungszeit
- Seltenere RAM-Zugriffe

Fazit

Vorteile

- Kürzere Ausführungszeit
- Seltenere RAM-Zugriffe
- Weniger Gesamt-Traffic

Fazit

Vorteile

- Kürzere Ausführungszeit
- Seltenere RAM-Zugriffe
- Weniger Gesamt-Traffic

Nachteile

- Höherer Traffic relativ zur Ausführungszeit

Fazit

Vorteile

- Kürzere Ausführungszeit
- Seltenere RAM-Zugriffe
- Weniger Gesamt-Traffic

Nachteile

- Höherer Traffic relativ zur Ausführungszeit
- Schlechtere Echtzeitfähigkeit

Fazit

Vorteile

- Kürzere Ausführungszeit
- Seltenere RAM-Zugriffe
- Weniger Gesamt-Traffic

Nachteile

- Höherer Traffic relativ zur Ausführungszeit
- Schlechtere Echtzeitfähigkeit

Unterschiede zu anderen Architekturen

- Kleinere Caches

Fazit

Vorteile

- Kürzere Ausführungszeit
- Seltenere RAM-Zugriffe
- Weniger Gesamt-Traffic

Nachteile

- Höherer Traffic relativ zur Ausführungszeit
- Schlechtere Echtzeitfähigkeit

Unterschiede zu anderen Architekturen

- Kleinere Caches
- Exklusiv

Fazit

Vorteile

- Kürzere Ausführungszeit
- Seltenere RAM-Zugriffe
- Weniger Gesamt-Traffic

Nachteile

- Höherer Traffic relativ zur Ausführungszeit
- Schlechtere Echtzeitfähigkeit

Unterschiede zu anderen Architekturen

- Kleinere Caches
- Exklusiv
- Kein getrennter Instruction-Cache