In-GPU-memory MOLAP aggregation using CUDA Dynamic Parallelism

Donnerstag, 23. Juli 2015

Abschlusskolloquium

UNI FREIBURG

Albert-Ludwigs-Universität Freiburg

Jérôme Meinke
Student im Studiengang BSc. Informatik

Albert-Ludwigs-Universität Freiburg
Jedox AG, Freiburg

Thema & Motivation

UNI FREIBURG

- **Thema:**
  - MOLAP-Aggregation mithilfe von GPUs
- **Kontext:**
  - In-memory OLAP-Server von Jedox AG
  - interaktives Mehrbenutzer-Szenario
  - Schreiben & Lesen zu jeder Zeit

    → „on-the-fly“-Berechnung
- **Motivation:**
  - Ist Beschleunigung auf GPU durch Funktion CDP möglich?
  - Wenn ja, wie?

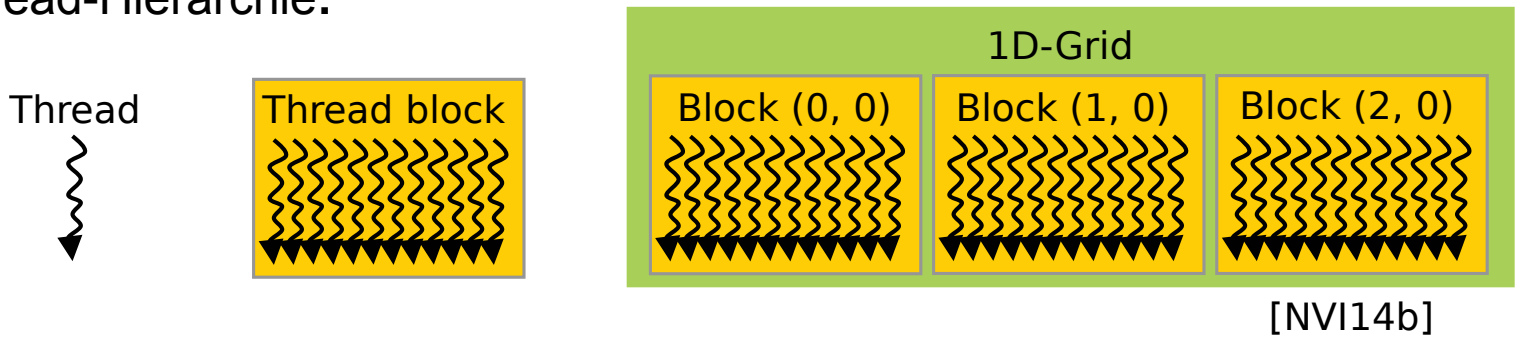
Übersicht

UNI FREIBURG

1. Technischer Hintergrund
    - GPU-Architektur und CUDA
    - MOLAP-Aggregation mithilfe von GPUs
    - CUDA Dynamic Parallelism
2. Einsatz von CDP
    - Probleme und Lösungen
3. StOAP
4. Testmethoden
5. Ergebnisse & Erkenntnisse
6. Future Work

GPU-Architektur und CUDA

- Begriffe: **host** (→ CPU) und **device** (→ GPU)
- CUDA-Kernel (sequentielles Programm) in **SIMT**-Kontext
  - 1 Warp → 32 Threads
  - Maskierung von aktiven und inaktiven Threads → **Divergenz**
- Thread-Hierarchie**:**

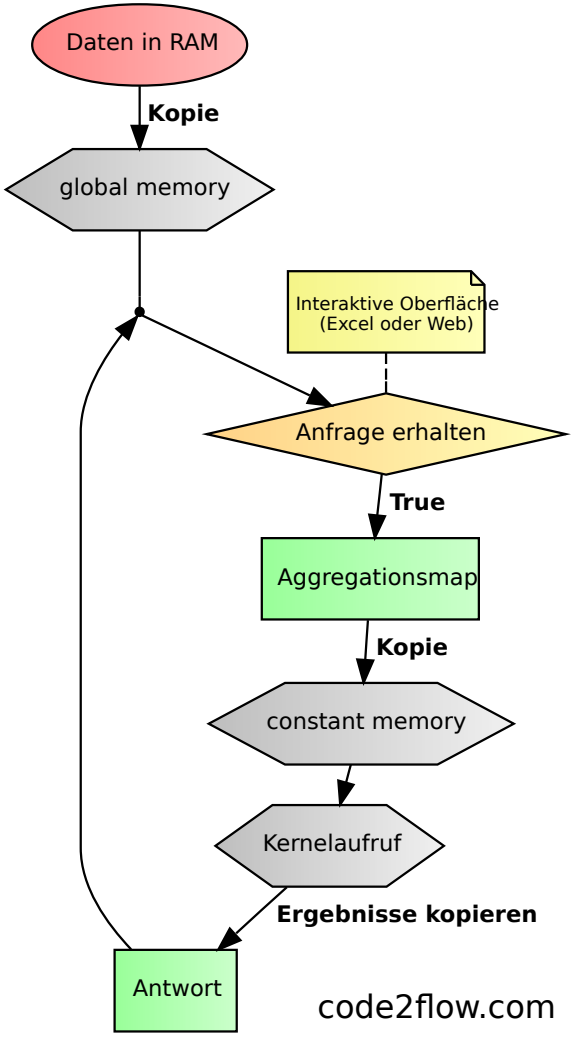


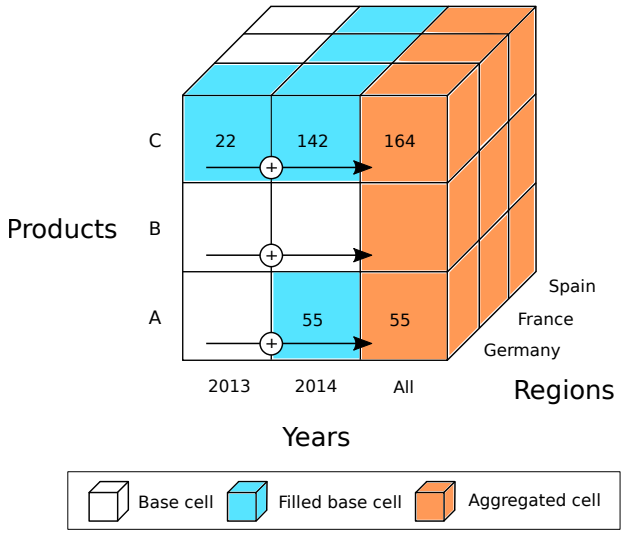
[NVI14b]

- Speicher-Unterteilung:
  - **Global**
  - **Constant**
  - **Shared** (+ Local)

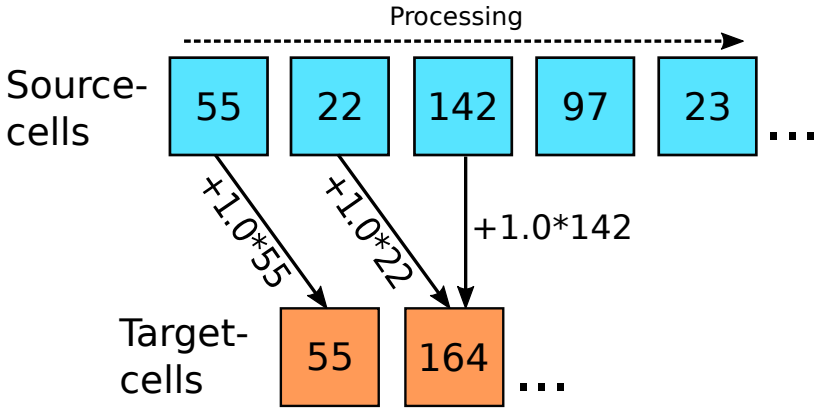
MOLAP-Aggregation mithilfe von GPUs

Ablauf bei Anfrageerhalt





Methoden: zielbasiert od. quellbasiert



CUDA Dynamic Parallelism (CDP)

UNI FREIBURG

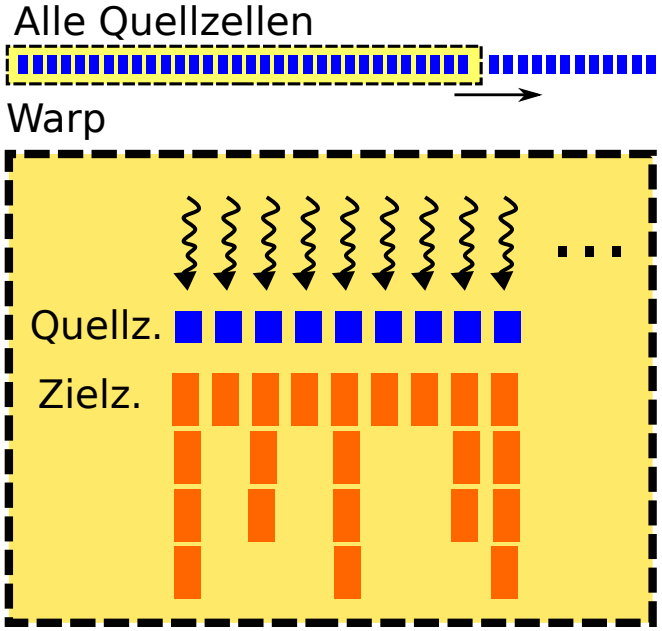
Möglichkeit, einen Kernel aus einem anderen zu starten

- **Vorteile**:
  - kein Umweg über ***host***
  - Ausnutzen von "dynamischem Parallelismus"

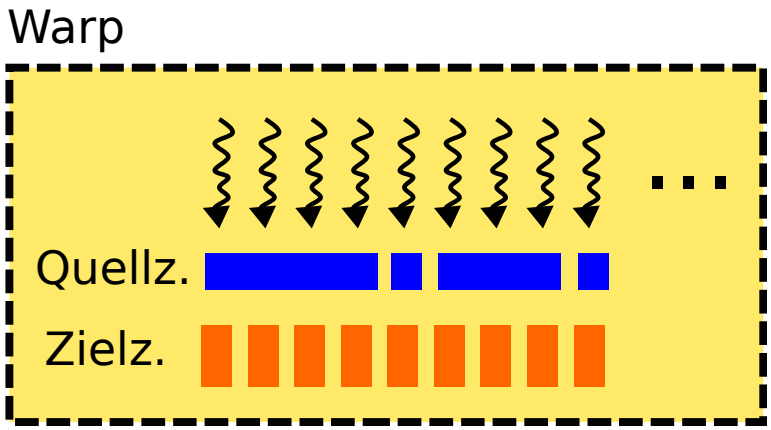
- **Nachteile**:
  - relativ viele Einschränkungen
  - Individuell: Kosten für Erkennen von zusätzlichem Parallelismus

Einsatz von CDP (1/2)

- Flaschenhals im urspr. Kernel:
  - Unterschiedlich viele *Zielzellen pro Quellzelle*...
    → „*idle*“ Threads innerhalb der Warps (**Divergenz**)
    → suboptimale Nutzung der Rechenleistung



Idee: Umverteilung der Arbeit in neuem Kernel → **Child**



Einsatz von CDP (2/2)

UNI FREIBURG

Einsatz von CDP und umverteiltes Schreiben der Zielzellen:

Problem	**Lösung**
Kernel-Aufruf-Kosten	Aufruf nur pro Parent-Block
Parent-Child Informationsfluss	Berechnung und Reservierung von Speicher für Zielzellzahlen
Parent-Child Ausführung	explizite Synchronisierung (CUDA Funktion) nach Child-Call
Addressierung der Quellzellen	Übergabe per Aufrufparameter + Berechnung
Auswahl Zielzelle im Child-Thread	**Upper-Bound-Search** in Prefix-Sum

StOAP

= Single-**t**hreaded **O**LAP **A**ggregation **P**rocessor:

- für fairen Vergleich von **CPU** (*sequentiell*) und **GPU** (*parallel*)
- lädt Jedox-Datenbanken (read-only)
- Code auf GitHub (github.com/jmeinke/StOAP)

- Google's dense hash map für Cube-Daten
  - schnell, aber spezialisierte DS für MOLAP besser
  - im Web keine **freien + dokumentierten** Implementierungen, die ohne Erweiterung benutzbar wären
- Optimierung mittels **Poor man's profiling**
  - einfach, trotzdem 5x schneller als vorher

Wie wurde getestet?

- **Test-Datenbanken und Anfragen:**
  - Demo-DB (281 Mio. Einträge)
  - Machines-DB (41 Mio. Einträge)
  - Anfragen: 3x versch. Bereichsgrößen (Klein, Mittel, Groß)
  - bei GPU: Preaggregation aktiv/inaktiv

- **Korrektheit der Ergebnisse:**
  - Anpassung Perl-Skript „PerfTest“ für StOAP (I/O)

- **Performance:**
  - Bash-Skripte für Senden von Requests
  - extra Log-Einträge im Server-Code
  - jeweils opt. Zahl paralleler Hashfunktionen (Serientests)

    → dann Durchschnitt von 10 Wdh.

Ergebnisse & Erkenntnisse

UNI FREIBURG

- **GPU vs. StOAP**: GPU 16x-218x schneller
- **allgemein gültige Aussagen schwierig**
  - Performance abhängig von Query, Cube-Struktur, #PHF, benutztem Tesla-Modell und Kernel-Scheduling

- **CDP-Auswirkung**:
  - ohne Preaggregation: „Mittel & Groß“ 22% schneller
  - mit Preaggregation: alle im Ø 42% langsamer
  - Spezialfall:
    - 364% bzw. 372% schneller (o/m Preaggregation)

Fazit: Methode hat Potenzial, aber eine Weiche fehlt noch

→ sie würde die **Dynamik** bringen, die in C**D**P enhalten ist

Future work

- **Möglichkeit 1:** für die Unterscheidung CDP ja/nein:
  - 1 Parent-Kernel + 2 verschiedene Child-Kernel
  - Parent-Kernel (***device***) stellt Varianz fest und wählt Child

- **Möglichkeit 2:** wie 1., aber mit nur einem Child
  (alte Funktionsweise zusätzlich im Parent)

- **Möglichkeit 3:**
  - Struktur letzter Dimension des Cubes relevant

    → Analyse der Anfrage ggü. Cube-Struktur

    → Entscheidung durch ***host***

Ende des Vortrags

UNI FREIBURG

Fragen?

Bibliographie & Quellenverzeichnis

- [Eic13] Susanne Eichel. “Parallele Berechnung großer spärlich besetzter aggregierter Bereiche mit Hilfe von Grafikprozessoren”. MA thesis. Albert-Ludwigs-Universität Freiburg im Breisgau, Sept. 2013
- [NVI13] NVIDIA. NVIDIA’s Next Generation CUDA Compute Architecture: Kepler GK110. Whitepaper. Jan. 2013. URL: http://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf
- [NVI14b] NVIDIA. CUDA C Programming Guide. PG-02829-001. Version 6.5. NVIDIA Corporation, Aug. 2014. URL: http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf

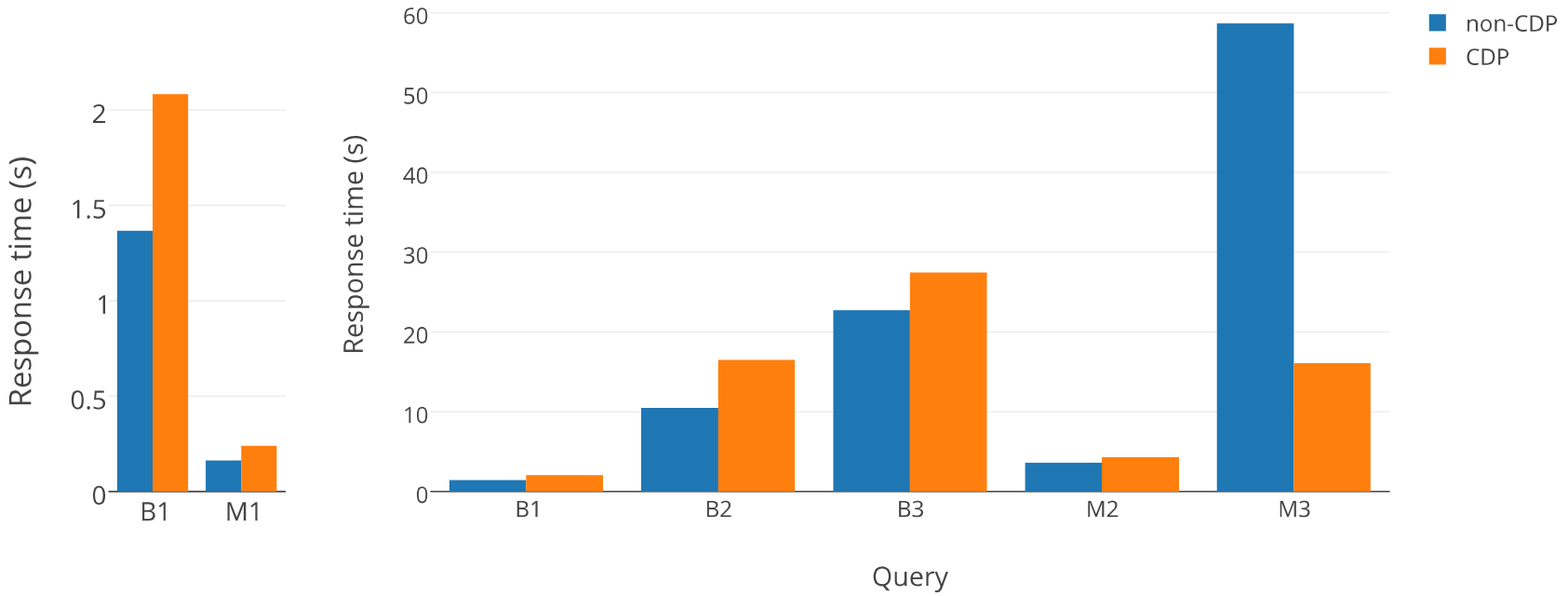
Infos zu Anfragen

UNI FREIBURG

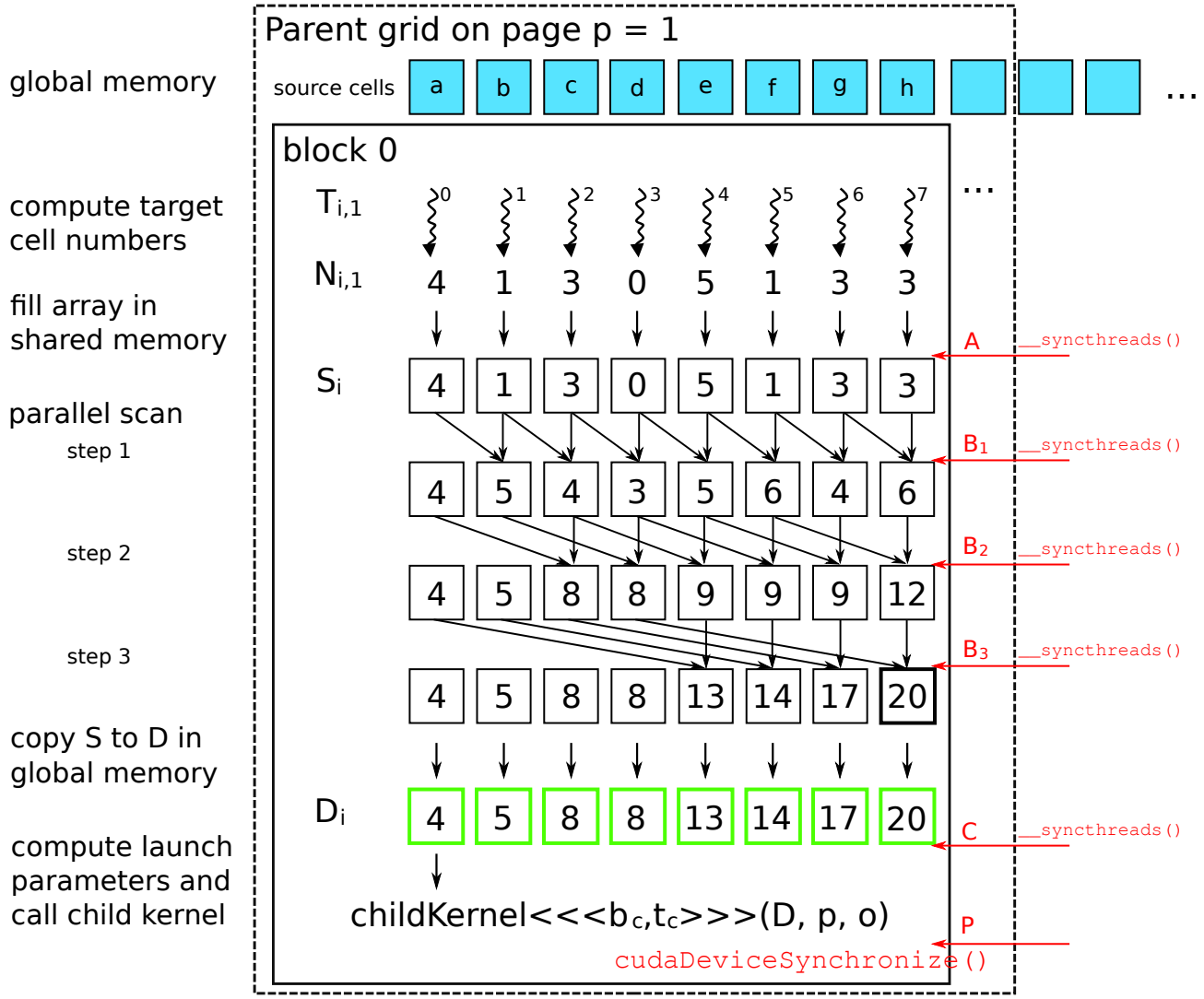
Table 5.1.: Test-query properties

Query	**Source cells** Relevant	**Target cells** Filled	Overall
B_1	281057088	1	1
B_2	281057088	228	228
B_3	281057088	17460	17460
M_1	41294400	1	1
M_2	6216000	2519	2519
M_3	29534400	13971	874437

Ausführungszeiten

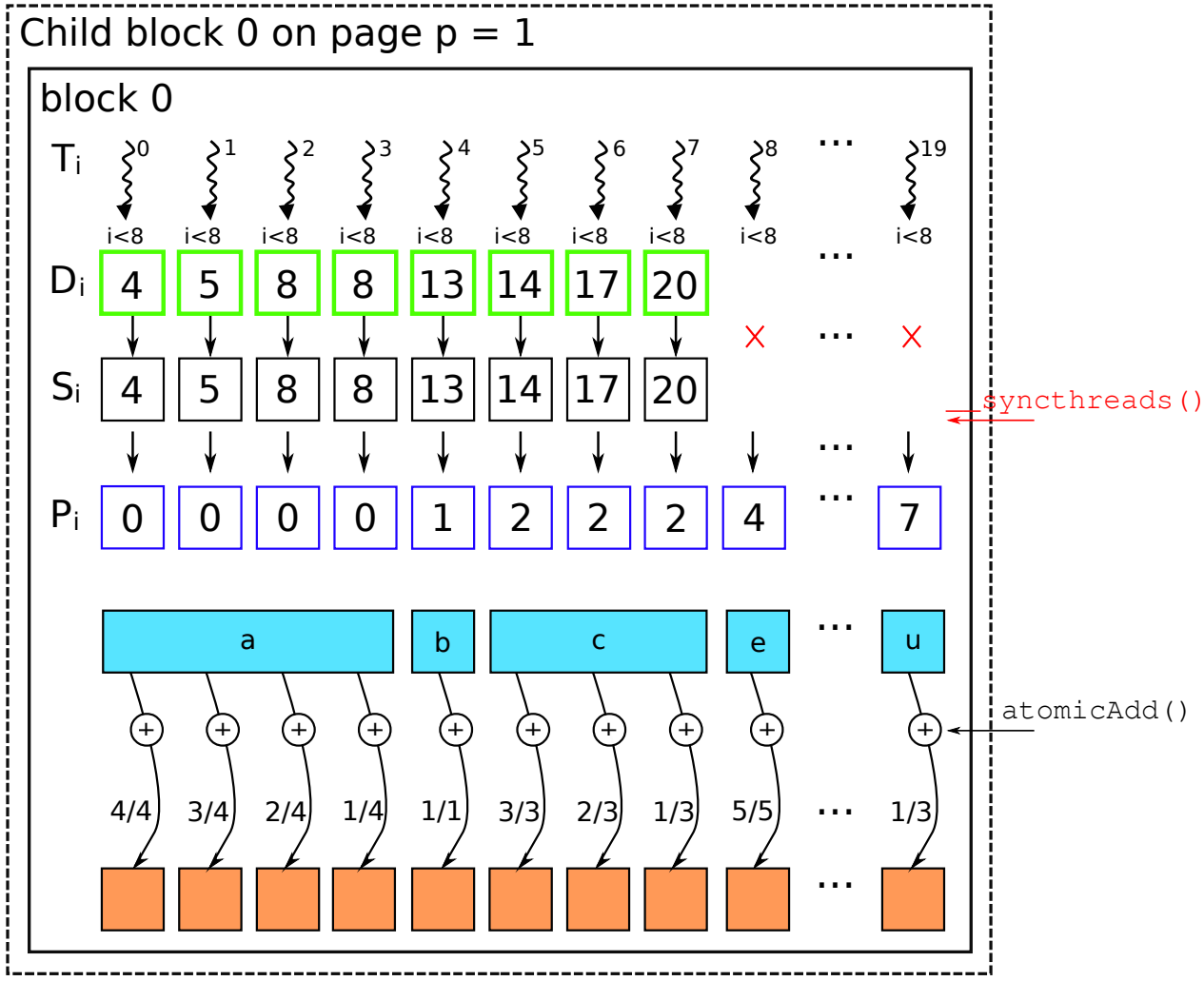


Beispiel: CDP-Kernel (Parent)



Variablen:
- T = Thread Index
- N = Anzahl Zielzellen
- S = Shared M. Array
- D = Global M. Array
- p = Page Index
- o = Page Offset

Beispiel: CDP-Kernel (Child)



copy D_i for i < b to shared mem

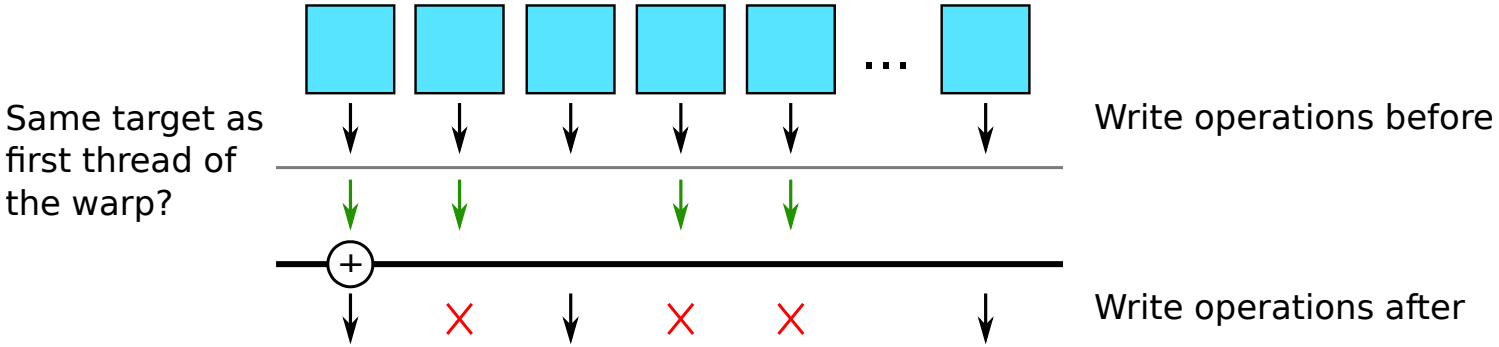
search for i in S yields thread id of the parent

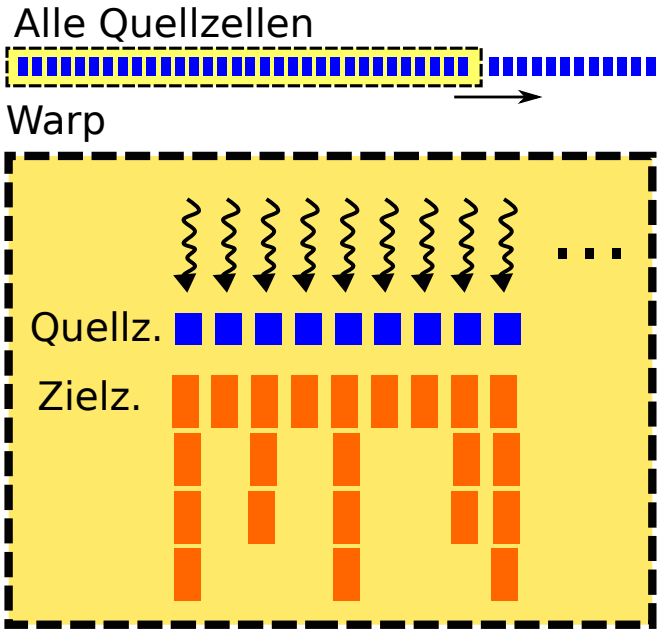
retrieve value of source cell P_i

get coordinates of the individual target cell S_{Pi} - i

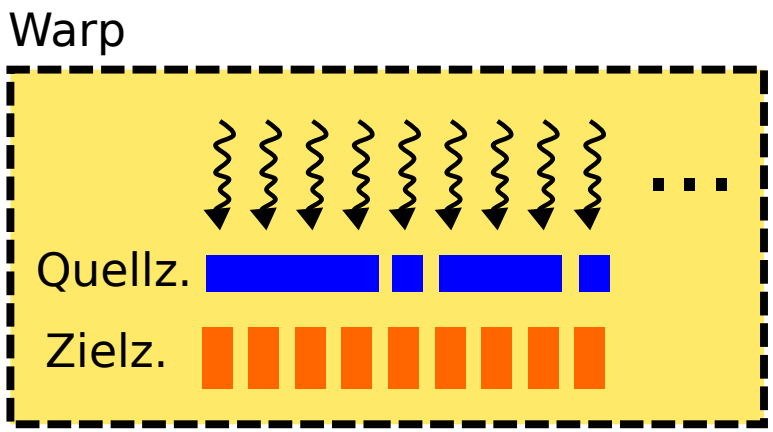
add value to the specified target

Urspr. Warp-Preaggregation



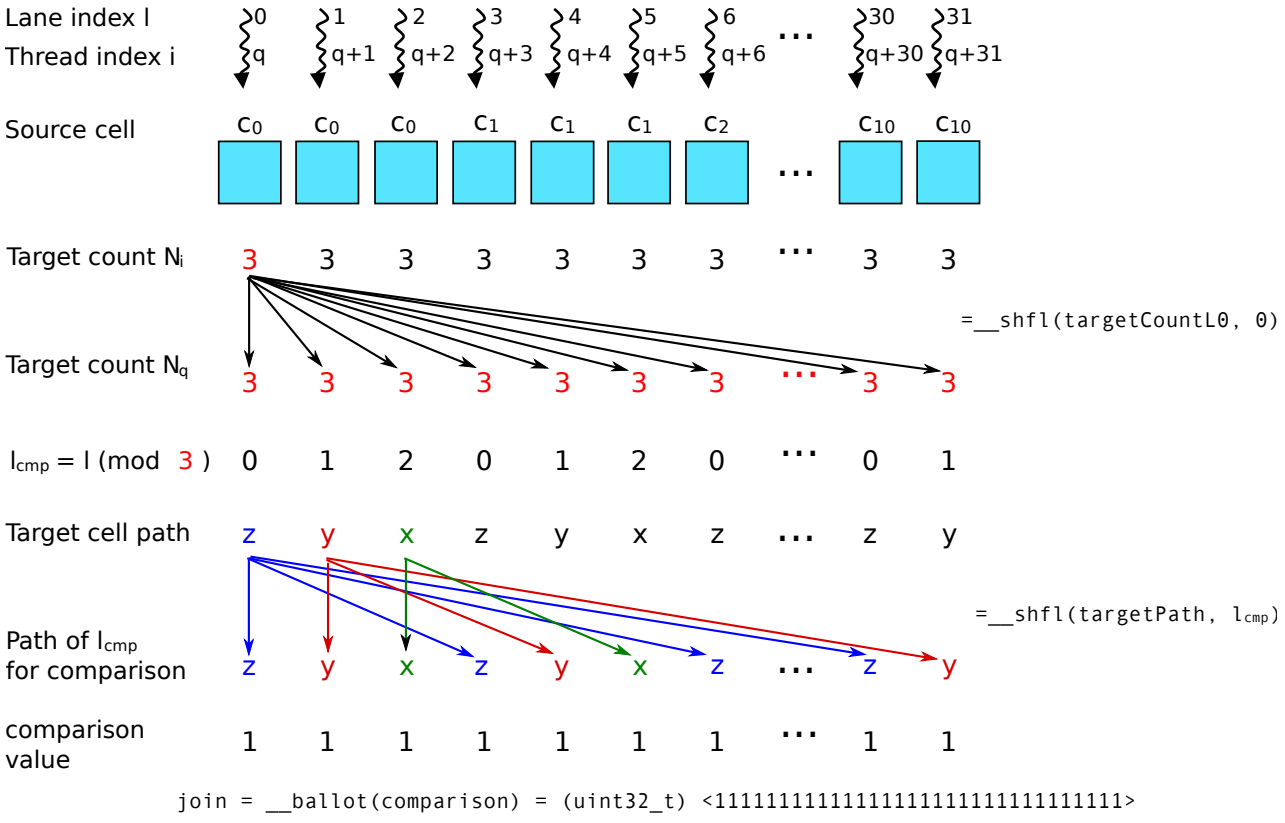


Greift in Child-Kernel nicht mehr

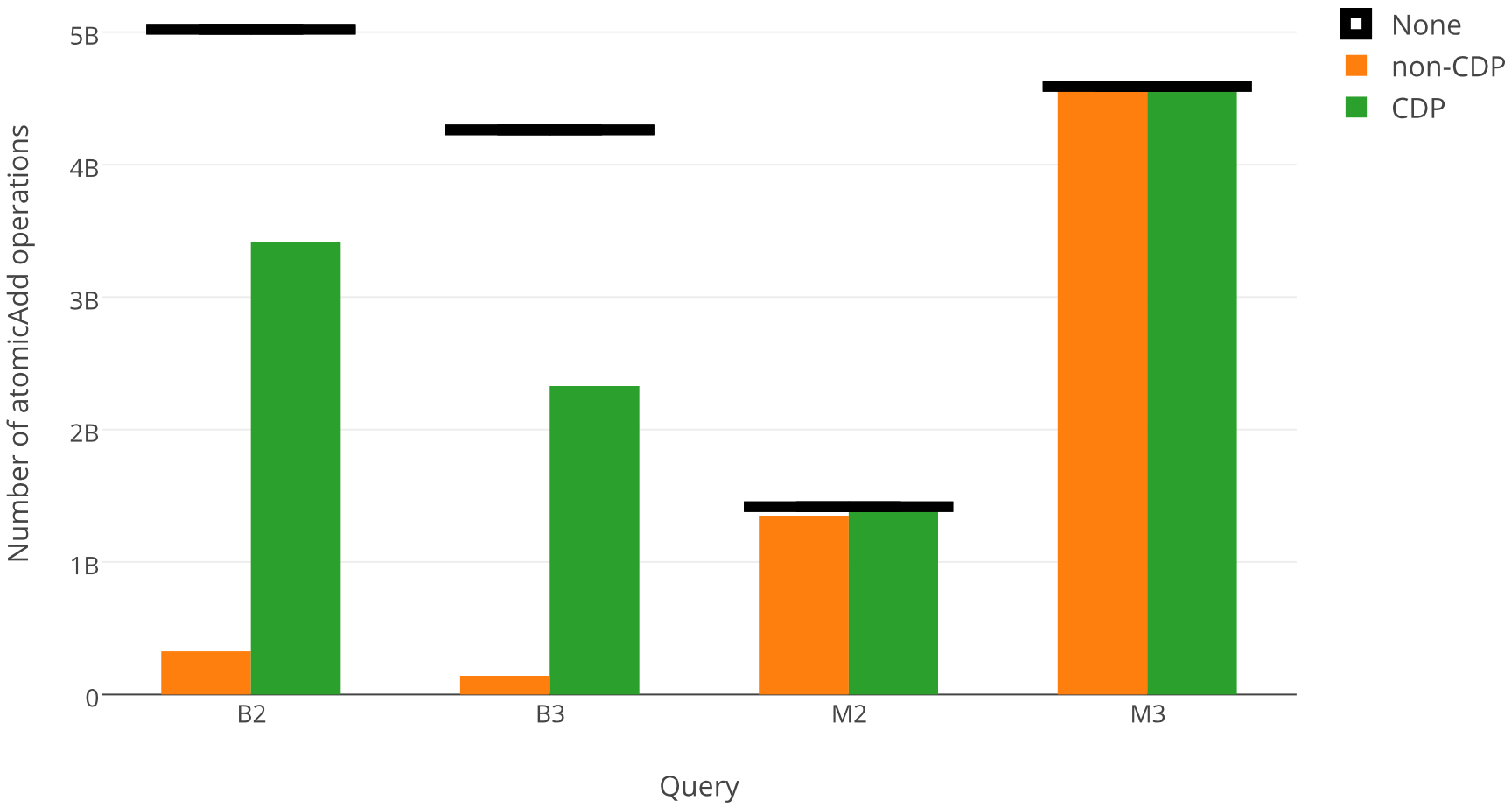


Warp-Preaggregation in Child

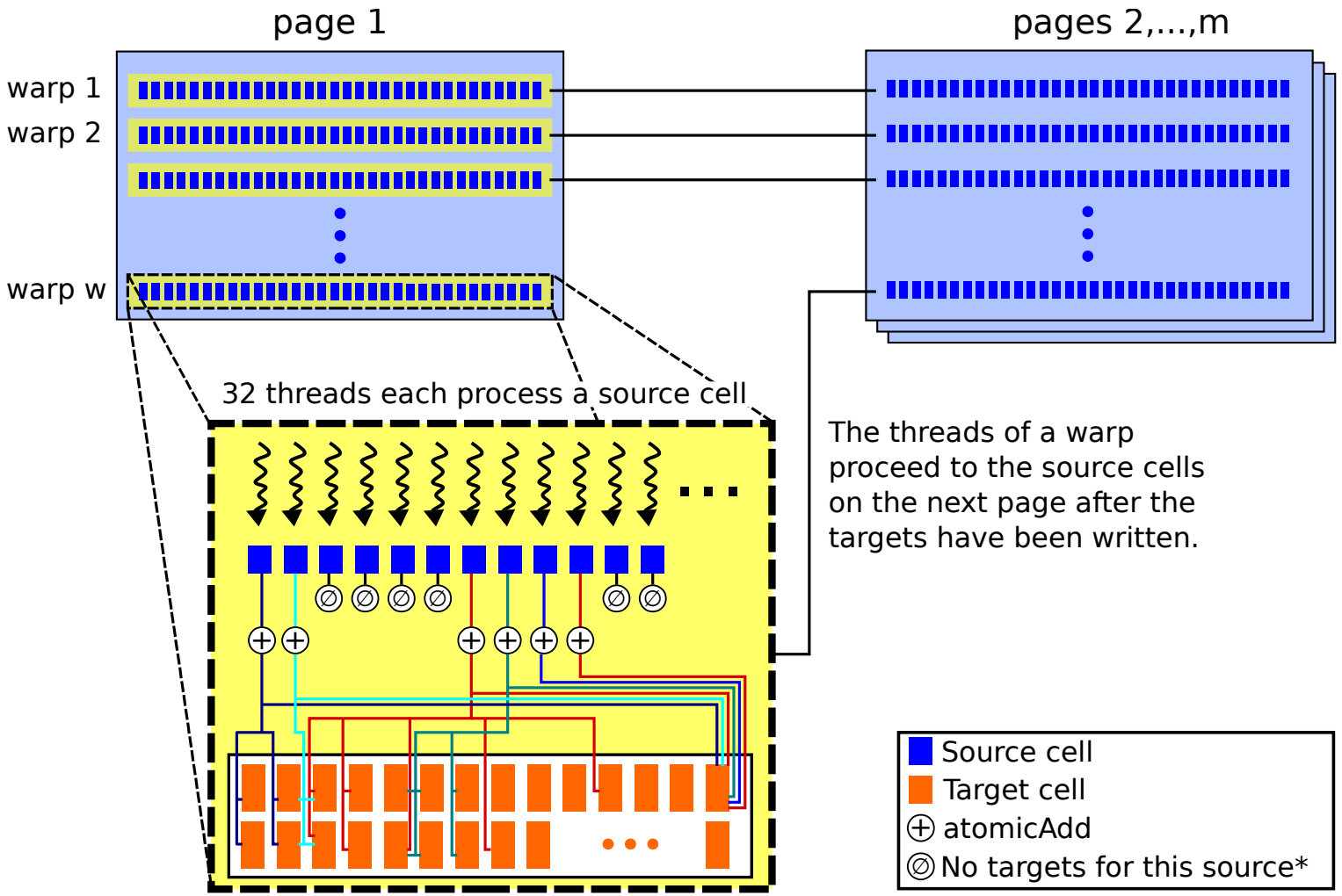
- Neue Methode: Warp-Preaggregation



Effizienz der Warp-Preaggregation

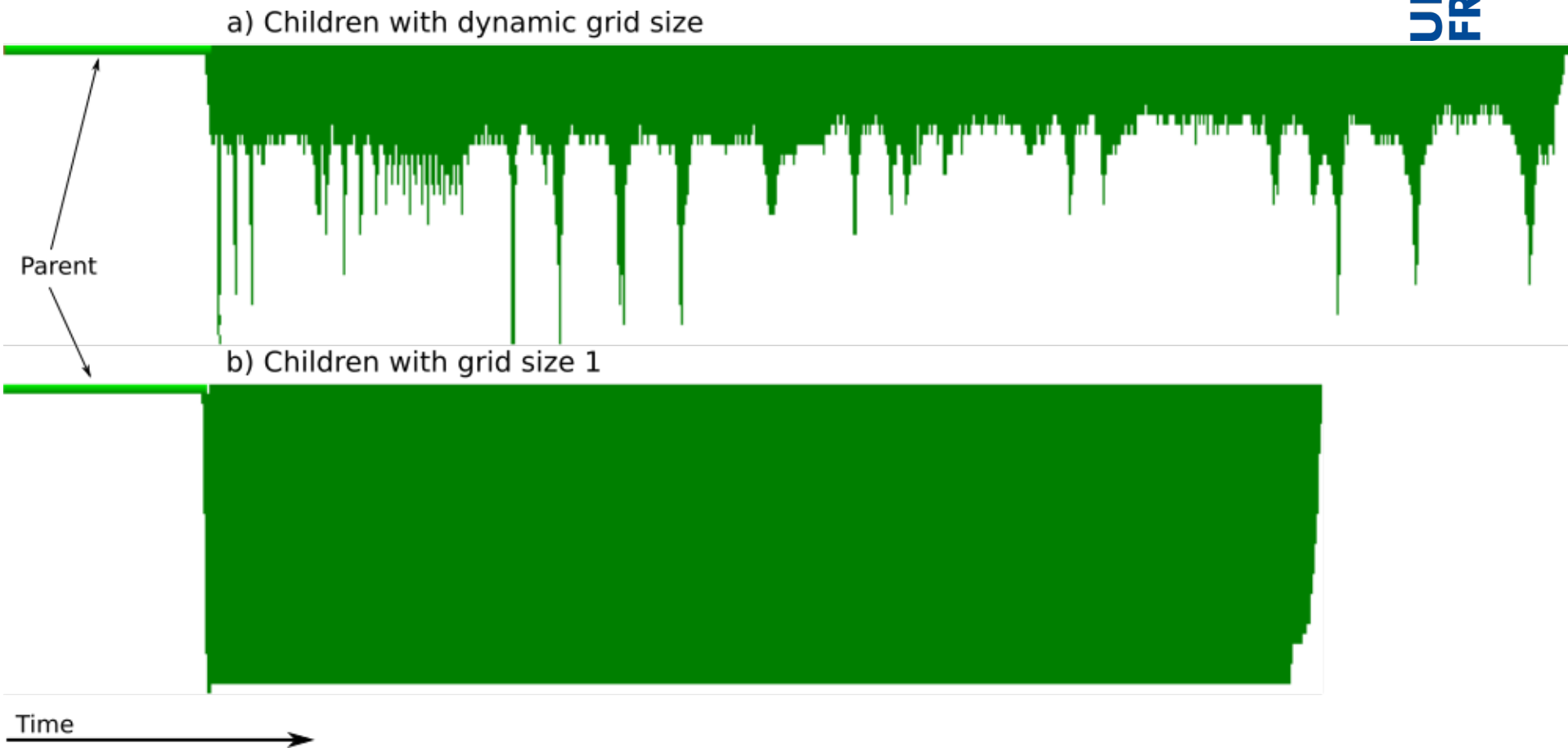


Ursprünglicher Kernel

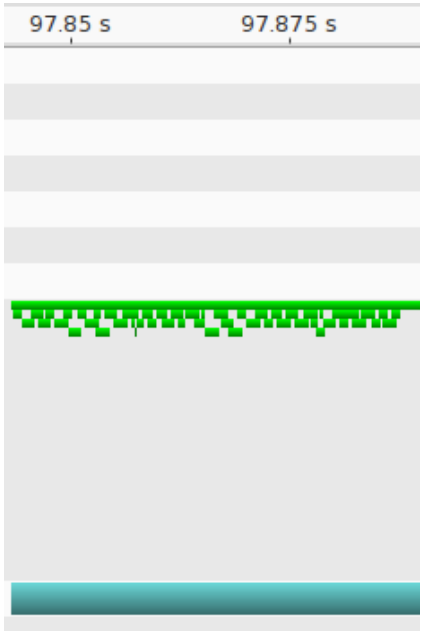


*Source cells without target cells only occur when the prefilter step is ommitted.

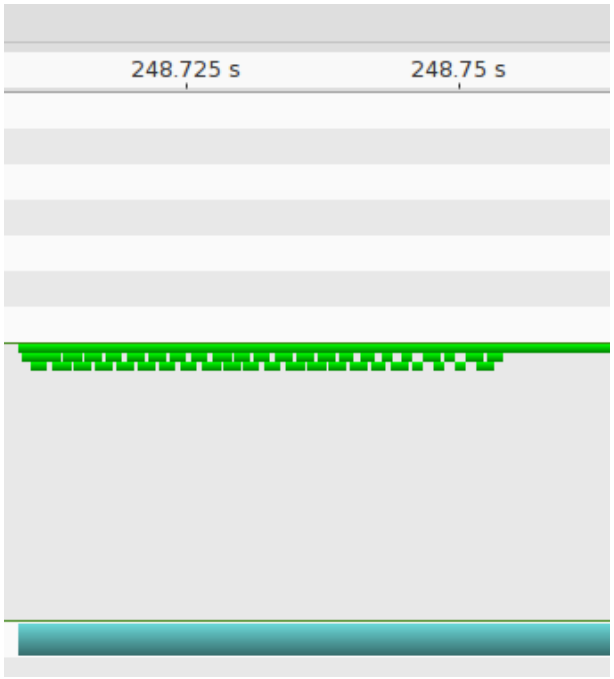
Visual profiler output (Grid sizes)



Visual profiler output (Grid sizes)



vs.



Child grid has multiple blocks

Child grid has only 1 block