technische universität
dortmund

Masterarbeit

Effiziente String-Verarbeitung in Datenbankanfragen auf hochgradig paralleler Hardware

Florian Lüdiger
Juni 2019

Gutachter:
Prof. Dr. Jens Teubner
Henning Funke

Technische Universität Dortmund
Fakultät für Informatik
Datenbanken und Informationssysteme (LS-6)
http://dbis.cs.tu-dortmund.de

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation und Hintergrund

1.2 Aufbau der Arbeit

Kapitel 2

Grundlagen der CUDA-Programmierung

Kapitel 3

Der Pipelining-Ansatz

Kapitel 4

Einfacher, paralleler String-Vergleich

Für die Evaluation des Lane-Refill-Verfahrens für die Verarbeitung von String-Daten wird zunächst ein einfacher String-Vergleich auf einer GPU untersucht. Ein Vergleich auf Gleichheit ist dabei die einfachste Variante von String-Verarbeitung, die vom Lane-Refill profitieren könnte. Diese Untersuchung wird dabei helfen, zu erfahren, ob die Anwendung des Lane-Refill-Verfahrens bei String-Daten allgemein Potenzial dafür bietet, den Durchsatz entsprechender Anwendungen zu erhöhen.

Zunächst wird dazu ein String-Vergleich mittels der CUDA Schnittstelle ohne spezielle Optimierungen implementiert, um einen Vergleich mit der optimierten Version durchführen zu können. Außerdem wird eine leichte Anpassung an dem Verfahren vorgenommen, sodass ein alternativer Workload für weitere Tests genutzt werden kann.

4.1 Vorgehen

Als Basis für die Untersuchung wird zunächst der Gleichheitstest für Strings naiv, also ohne tiefgehende Optimierungen umgesetzt.

4.2 Implementierung

```
1  __global__
2  void naiveKernel(
3  int *char_offset,          // indices of the first letter of every string
4  char *data_content,        // concatenated list of compare strings
5  char *search_string,       // string that will be searched for
6  int search_length,         // length of the search string
7  int line_count,            // number of lines in the data set
8  int *number_of_matches) { // return value for the number of matches
9
10   // global index of the current thread,
11   // used as the iterator in this case
12   unsigned loop_var = ((blockIdx.x * blockDim.x) + threadIdx.x);
13
14   // offset for the next element to be computed
15   unsigned step = (blockDim.x * gridDim.x);
16
17   bool active = true;
18   bool flush_pipeline = 0;
19
20   while(!flush_pipeline) {
21     // element index must not be higher than line count
22     active = loop_var < line_count;
23
24     // break computation when every line is finished and therefore inactive
25     flush_pipeline = !__ballot_sync(ALL_LANES, active);
26
27     data_length = char_offset[loop_var+1] - char_offset[loop_var] - 1;
28
29     // if the lengths of the strings don't match,
30     // the string can be discarded immediately
31     if (active && data_length != search_length)
32       active = false;
33
34     int search_id = 0;
35
36     // iterate over both strings till the end
37     // or until a non-matching character has been found
38       while(__any_sync(0xFFFFFFFF, active) && search_id < search_length) {
39
40       int data_id = search_id + char_offset[loop_var];
41
42       // when strings don't match, inactivate the lane
43       if (active && data_content[data_id] != search_string[search_id])
44         active = false;
45
46       search_id++;
```

```
47      }
48
49      // when comparison finishes without being inactivated,
50      // a match has been found
51      if (active)
52      atomicAdd(number_of_matches, 1);
53
54      loop_var += step;
55    }
56  }
```

Listing 4.1: Naive Implementierung des String-Vergleichs

4.3 Präfixtest als alternativer Workload

Kapitel 5

Das Lane-Refill Verfahren

Kapitel 6

Verbesserung des einfachen String-Vergleichs

6.1 Ansatzpunkte für Lane-Refill

6.2 Umsetzung mit Lane-Refill

Kapitel 7

Grundlagen von regulären Ausdrücken

Kapitel 8

Paralleler Musterabgleich mit regulären Ausdrücken

8.1 Vorgehen

8.2 Implementierung

Kapitel 9

Verbesserung des Verfahrens zum Musterabgleich

9.1 Ansatzpunkte für Lane-Refill

9.2 Umsetzung mit Lane-Refill

Kapitel 10

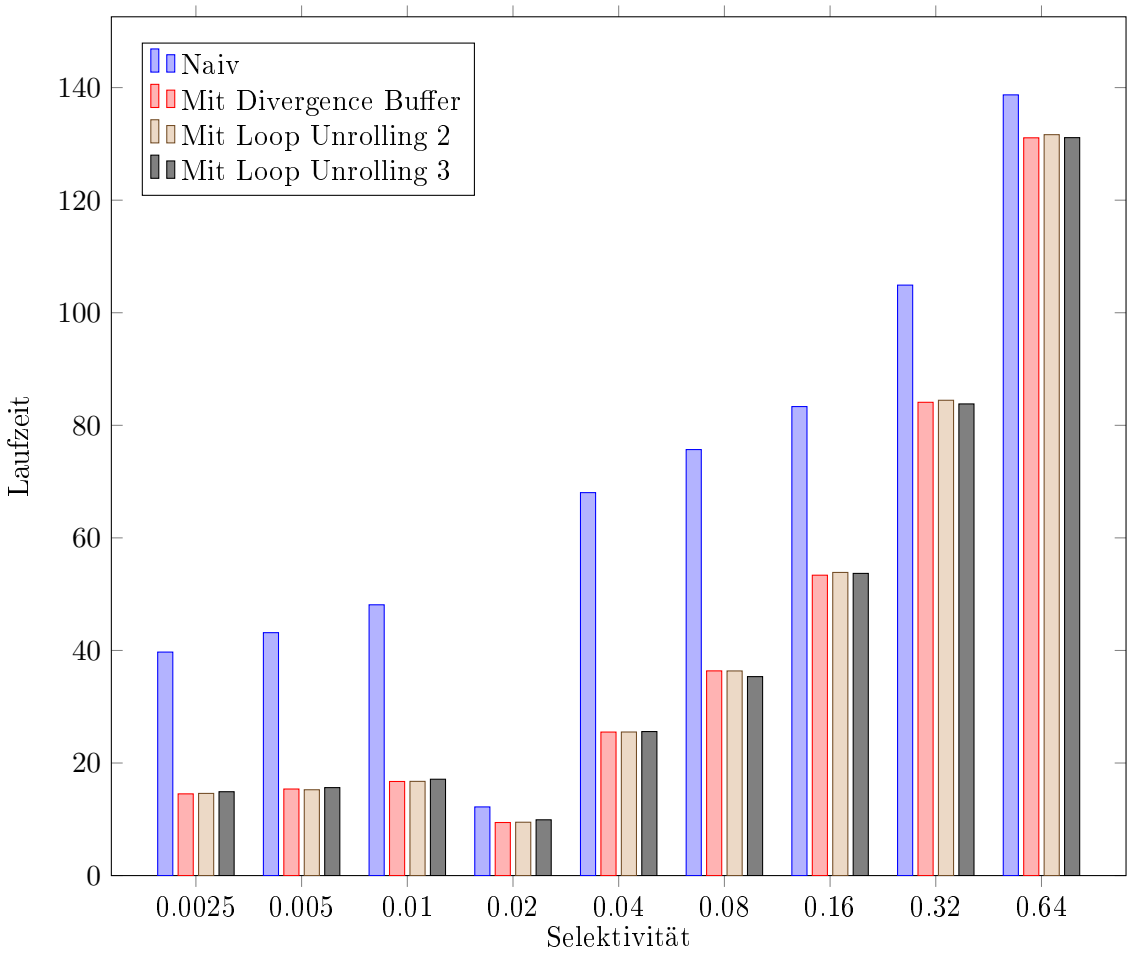
Optimierung der Ausführungsparameter

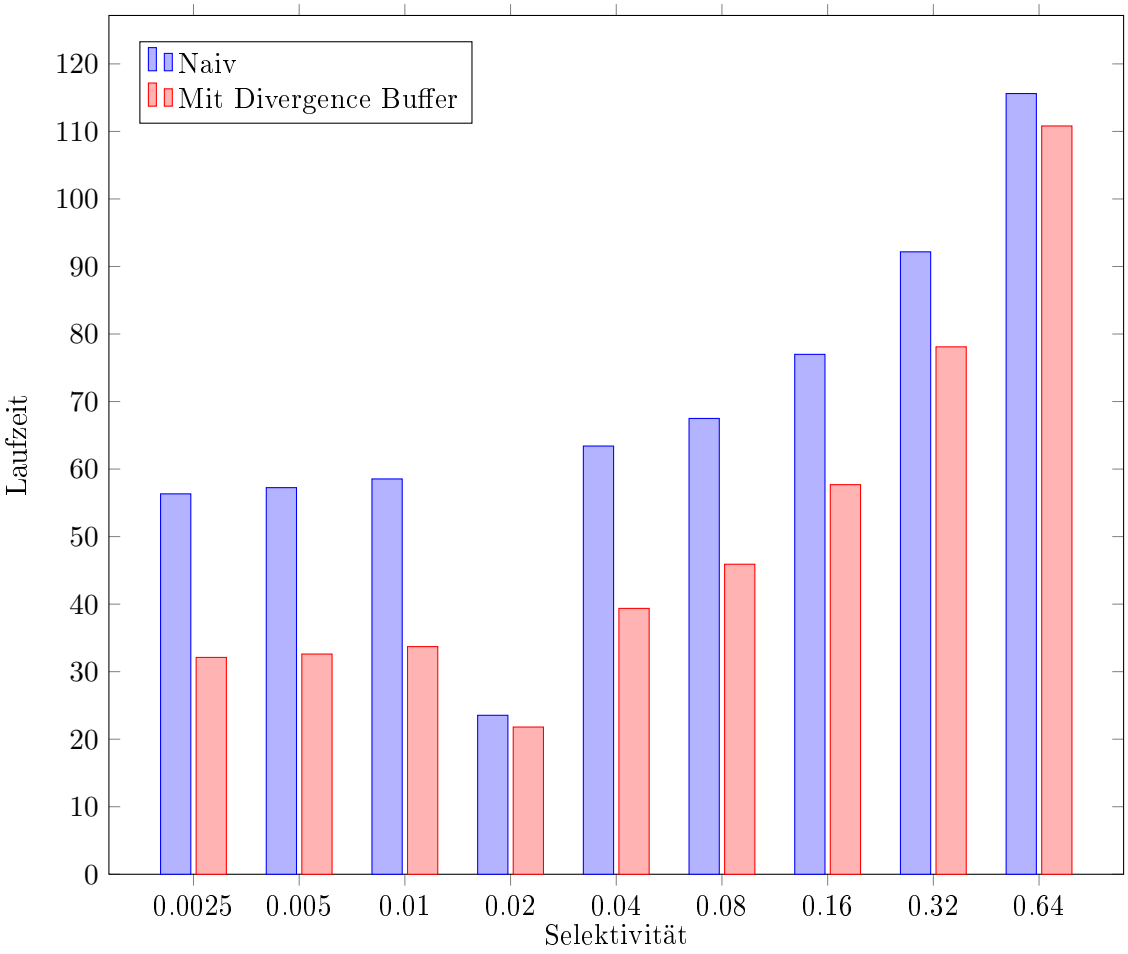
Kapitel 11

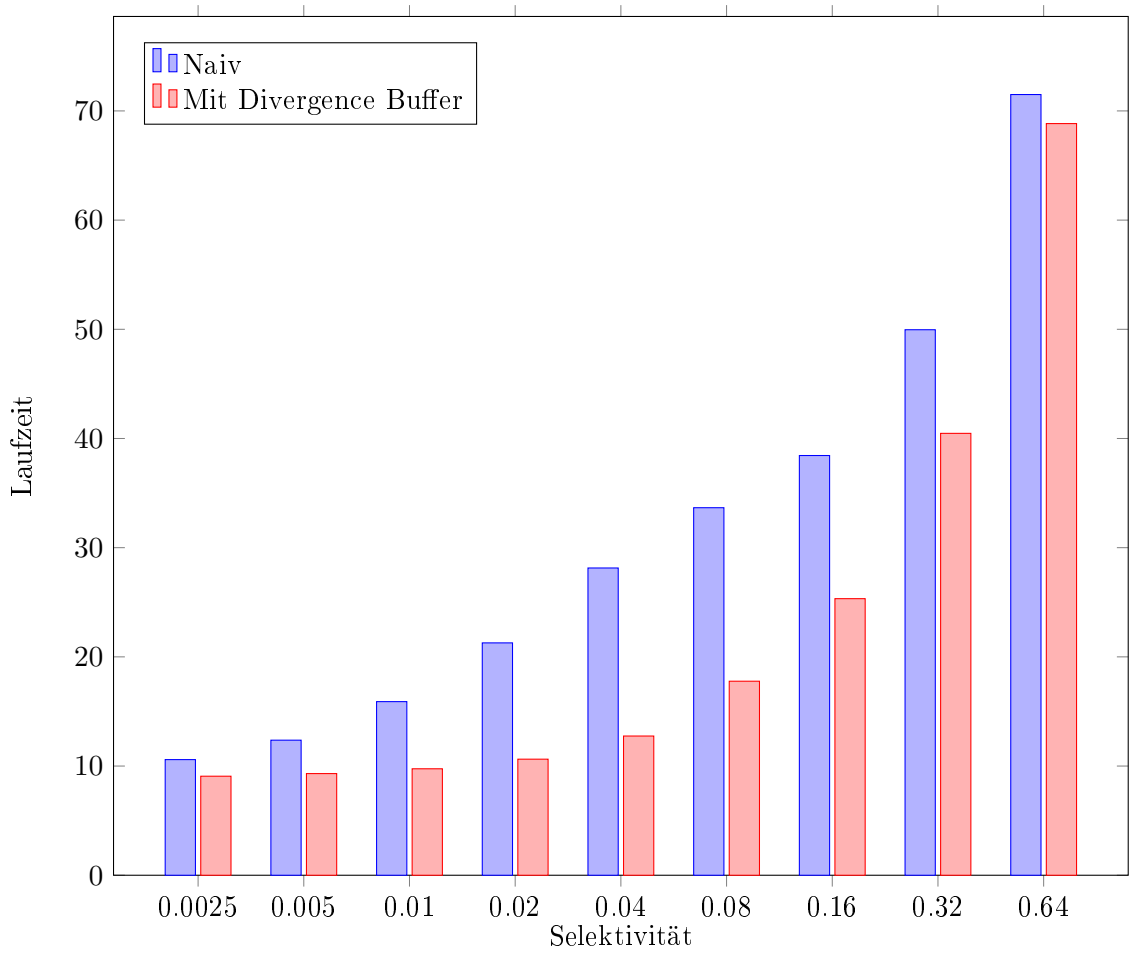
Evaluation des einfachen String-Vergleichs

11.1 Verwendete Workloads und deren Merkmale

11.2 Vorstellung der Messergebnisse




Naiv
Mit Divergence Buffer
Laufzeit
0
10
20
30
40
50
60
70
80
90
100
110
120
0.0025
0.005
0.01
0.02
0.04
0.08
0.16
0.32
0.64
Selektivität



11.3 Diskussion der Ergebnisse

Kapitel 12

Evaluation des parallelen Musterabgleichs

12.1 Verwendete Workloads und deren Merkmale

12.2 Vorstellung der Messergebnisse

12.3 Diskussion der Ergebnisse

Kapitel 13

Ergebnis und Fazit

Anhang A

Weitere Informationen

Abbildungsverzeichnis

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie Zitate kenntlich gemacht habe.

Dortmund, den 1. Februar 2019

Florian Lüdiger