TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik, Institut für Technische Informatik, Professur für VLSI-Entwurfssysteme, Diagnostik und Architektur

STUDIENARBEIT

Übertragung eines an MapReduce orientierten k-Means-Algorithmus auf einen FPGA-Hardwarebeschleuniger für den Cloud-Einsatz

Martin Knöfel
martin.knoefel@tu-dresden.de

Dresden, 16.03.2017

Überblick

Motivation

Literatur

Probleme

Zusammenfassung und Ausblick

01 Motivation
MapReduce Modelling

- Programmiermodell für große Daten
- Nutzung teilbarer Datensätze für Parallelisierung
- drei Schritte:

MAP:	Zuordnung der Daten mittels Schlüssel
SHUFFLE:	Verteilung der Daten auf Recheneinheiten
REDUCE:	Zusammenfassen der Teilergebnisse

- hohe Parallelisierung für Nutzung einer FPGA-Implementierung

01 Motivation
k-Means Clustering

Allgemein:

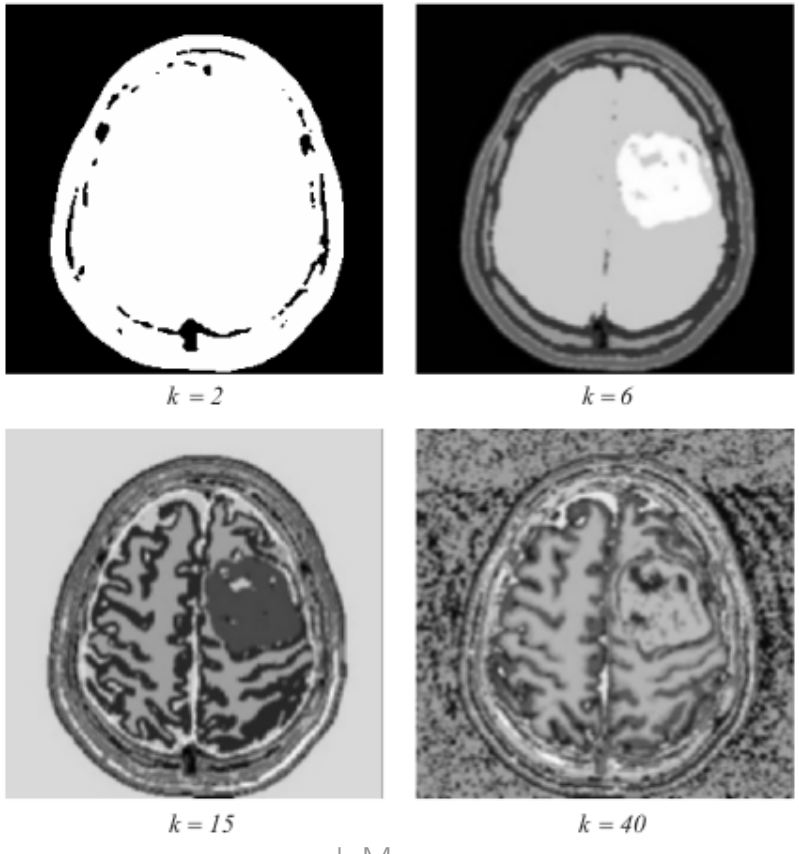
- Aufteilung von Daten in k-Cluster
- mittels “Abstandsberechnung” und “Zuordnungsmethode”
- NP komplex
- Standard-Algorithmus: Lloyd-Algorithmus

01 Motivation
Anwendung

- Signalverarbeitung
  - Farbquantisierung
  - Bildsegmentierung (z.B. in der Medizin)
- Maschinelles Lernen
  - Feature learning
    (zum Extrahieren von Merkmalen)
- Clusternanalysen
  - Marktanalysen
  - Wetter
  - Astronomy
  - Kartographie

01 Motivation

Anwendung



[1]

02 Literatur

Author	Typ	Speedup	
Choi et al.[2]	batch	20	against software
Saegusa et al.[3]	semi	7-9	realtime clustering of images
Asano et al.[4]	stream	7-12	against one thread on CPU
Jia et. al.[5]	stream	ca. 10-21	VHDL Implementierung with axi-stream
Lavenier[6]	stream	-	depends on Hardware
Winterstein[7]	-	-	compares HLS with VHDL solutions

03 Probleme

Fragen:

- Wie erstelle ich die Vergleichswerte?
- VHDL oder High-Level-Synthese?
- Wie soll die Ein -und Ausgabe aussehen?
- Wo sind die Optimierungsmöglichkeiten?

03 Problem: Vergleichswerte

Lloyd's-Algorithmus:

- **Zuordnung:**
  - berechne den Abstand zu allen Clusterzentren (euklidische Distanz)
  - Zuordnung zum Zentrum mit dem kleinsten Abstand (ähnlich: Methode der kleinsten Quadrate)
- **Anpassung:**
  - berechne die neuen Mittelwerte jedes Clusterzentrums
  - vergleiche ob sich die Zentren verschoben haben

03 Problem: Vergleichswerte

- Dimensionierung des Programmes
  - variable Datengröße
  - variable Vektorgröße
- Bildsegmentierung als Beispiel
  - geringe Clusteranzahl
  - RGB-Farbraum
  - Zuordnung der Pixel in Cluster
- Visualisierung der Ergebnisse

03 Problem: Vergleichswerte

Original

03 Problem: Vergleichswerte

k=0

03 Problem: Vergleichswerte

k=1

03 Problem: Vergleichswerte

k=2

03 Problem: Vergleichswerte

Zusammenstellung der Farbzentren

03 Problem: VHDL oder HLS?

- VHDL bekannt
- HLS: Vivado High Level Synthesis
- Vorteile/Nachteile der HLS?
  (Recherche über HLS-Tutorials)
- Welche Implementierung bietet sich an?

03 Problem: VHDL oder HLS?
VHDL

Vorteile:

- Beschreibungssprache (Abstraktion auf Schaltungsverhalten)
- gezieltes Design möglich (ASIC-Entwurf)
- Simulation über Testbenches

Nachteile:

- Fachwissen notwendig
- Beschreibung ist aufwändig
- Optimierungen bzgl. Parallelität schwierig

03 Problem: VHDL oder HLS?
Vivado HLS

Vorteile:

- C/C++ (bekannte Programmiersprache)
- hohe Abstraktionsebene (Algorithmenbeschreibung)
- automatische Synthetisierung (viele Optimierungsmöglichkeiten durch Direktiven)
- Tests in C/C++

Nachteile:

- spezifisches Wissen des Syntheseprogramms notwendig
- Vorgang der Synthese unbekannt

03 erstes Ergebnis

Vivado HLS - unoptimiert

I/O-Ports	Bit
centroids	8
vector	8
ncentroids	8
assignment	16
insgesamt:	40

Clock	ns	MHz
target	5,00	200
estimated	4,06	246
Uncertainy	0,62	

- **Intervall: ca. 1,3s**
- Python: 4,34s
- C++: 0,28s

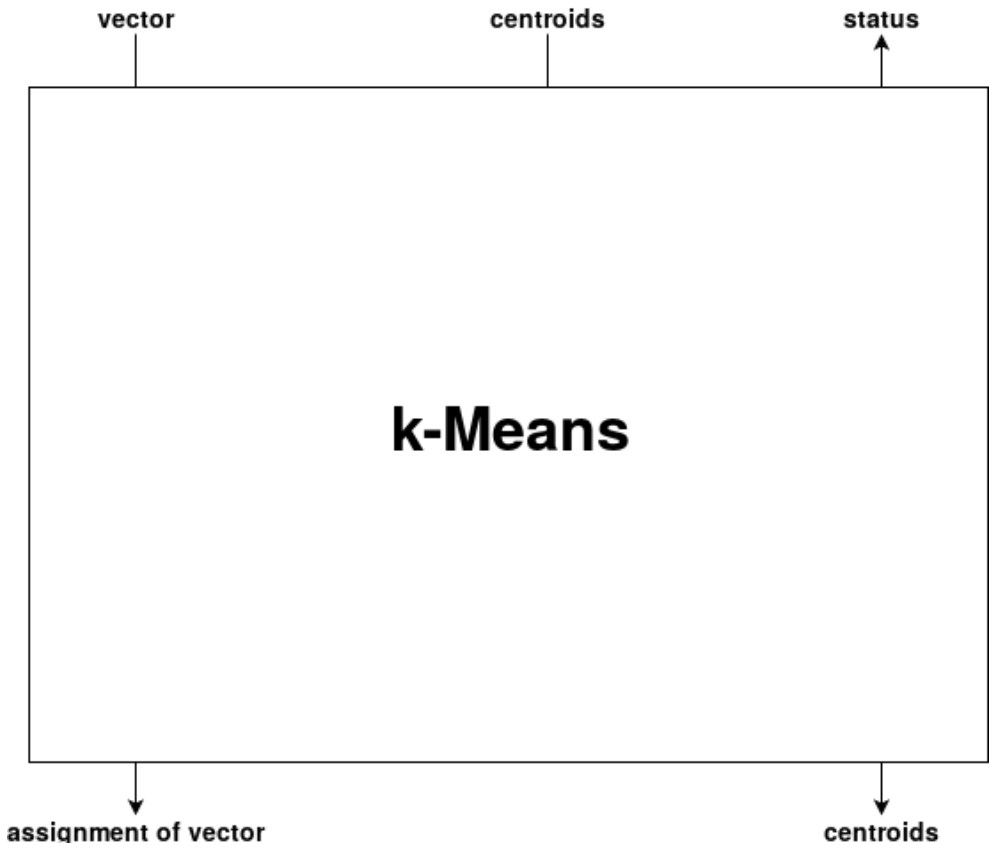
03 erstes Ergebnis

Vivado HLS - unoptimiert

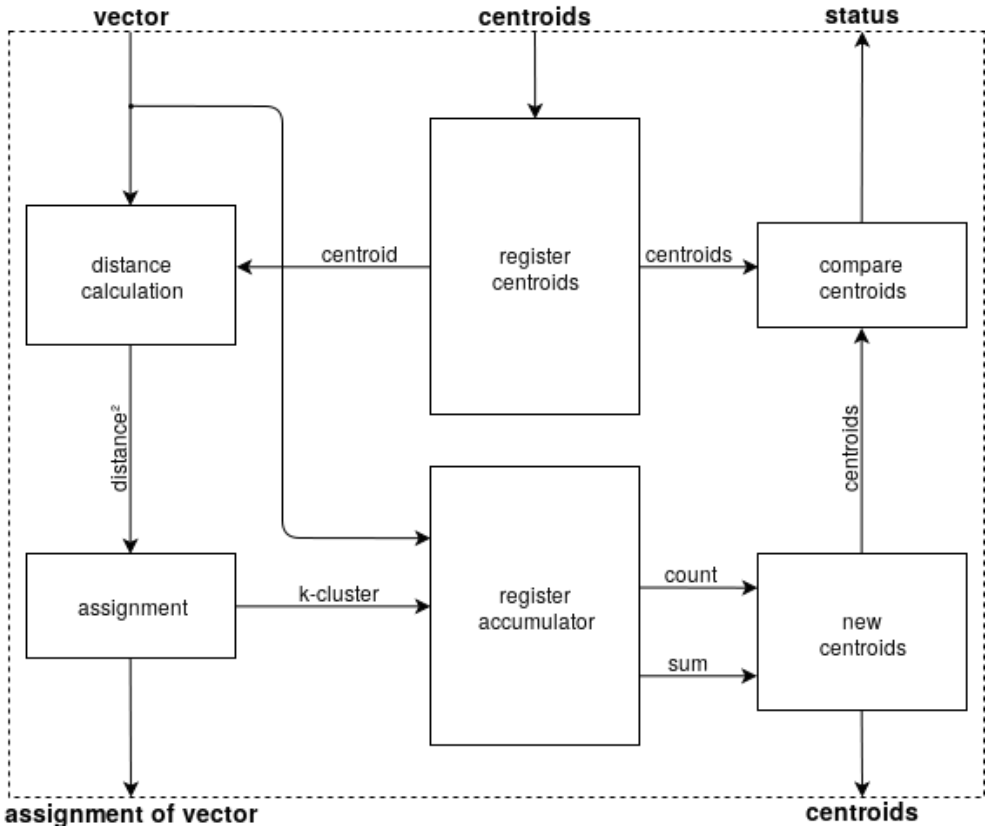
FPGA: VIRTEX-7 485T

	total	available	%
BRAM 18Kb	0	2060	0
DSP48E	11	2800	0,4
FF	1770	607200	0,3
LUT	2361	303600	0,8

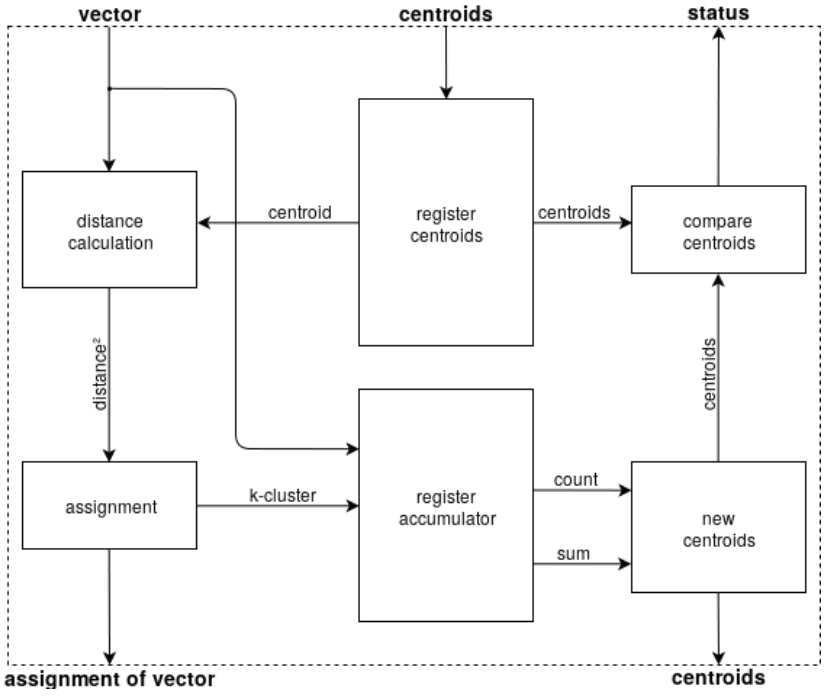
03 Problem: Ein -und Ausgabe



03 Problem: Optimierungsmöglichkeiten



03 Optimierungsmöglichkeiten



Optimierungsmöglichkeiten:

- assignment parallelisieren
- distance-Berechnung parallelisieren
- Assignment und Update voneinander trennen (rekonfigurierbare FPGA - für sehr große Datensätze)
- Pipelining (distance - assignment)
- datastreaming implementieren
- fertige IP-Cores parallel verwenden

04 Zusammenfassung und Ausblick

Idee:

- einfache Kernstruktur mit HLS erzeugen
- HLS nutzen um viele Optimierungen zu vergleichen

Vorerst:

- Festlegung auf Anwendung: Bildsegmentierung

Ausblick

- ein oder mehr IP-Cores für verschiedene Anwendungen
- Vergleich verschiedener Optimierungsmöglichkeiten
- Hardware-Tests der IP-Cores

Quellen:

[1] Heinz Handels, Siegfried Pöppl: *Medizinische Bildverarbeitung 2.Auflage*. 2009

[2] Yuk-Ming Choi, Hayden Kwok-Hay So: *Map-reduce processing of K-means algorithm with FPGA-accelerated computer cluster*. 2014

[3] Takashi Saegusa, Tsutomu Maruyama: *Performance comparison of FPGA, GPU and CPU in image processing*. 2009

[4] Shuichi Asano, Tsutomu Maruyama and Yoshiki Yamaguchi: *An FPGA implementation of real-time K-means clustering for color images*. 2007

[5] Fahui Jia, Chao Wang, Xi Li and Xuehai Zhou: *SAKMA: Specialized FPGA-Based Accelerator Architecture for Data-Intensive K-Means Algorithms*. 2015

[6] Dominique Lavenier: *FPGA implementation of the k-means clustering algorithm for hyperspectral images*. 2000

[7] Felix Winterstein: *High-Level Synthesis of Dynamic Data Structures: A Case Study Using Vivado HLS*. 2013