TECHNISCHE UNIVERSITÄT DRESDEN

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

Implementierung des Genom-Alignments auf modernen hochparallelen Plattformen

Vortrag zum Diplom

Oliver Knodel - Oliver.Knodel@mailbox.tu-dresden.de

Dresden, 08.03.2011

01 Einleitung

Problemstellung

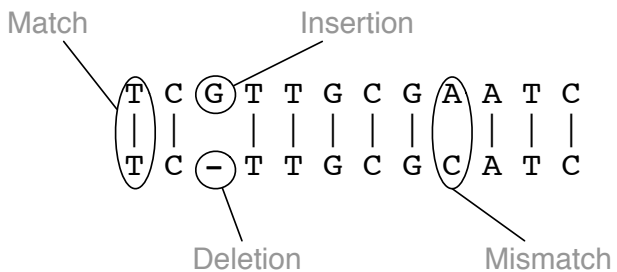
1. Neu gewonnene DNA-Sequenzen werden als erstes mit den bekannten Datenbanken abgeglichen, um Eigenschaften zu ermitteln
   - Inexakte Suche
   - Ergebnis ist die Position in der Datenbank
2. Exponentielles Wachstum der Datenbanken (mehrere Milliarden Basenpaare)
3. Neue kostengünstige und hoch parallele Sequenzierungsverfahren haben in den letzten Jahren zum *Short Read Mapping Problem* [TS09] geführt
   - Kurze Sequenzen (*Reads*) von 30 - 100 Basenpaaren
   - Ein Durchlauf von 2-3 Tagen erzeugt mehrere Millionen Reads
   - Suche mit großen Computer-Clustern dauert mehrere Tage und solche Systeme sind nicht für jeden verfügbar

$\Rightarrow$ **Schnelle kostengünstige Lösungen erforderlich**

02 Alignment-Algorithmen

Alignment-Algorithmen passen zwei Sequenzen aneinander an und liefern einen Wert (*Score*), der etwas über ihre Ähnlichkeit aussagt

- Um einen hohen Score zu erreichen, werden *Mismatches* und *Gaps* eingefügt
- Die Algorithmen können für die inexakte Suche in Datenbanken eingesetzt werden
  - Ergebnis einer Datenbanksuche ist die Position, an der ein bestimmter Schwellwert für den Score überschritten wird
  - Schwellwert muss vor der Suche festgelegt werden

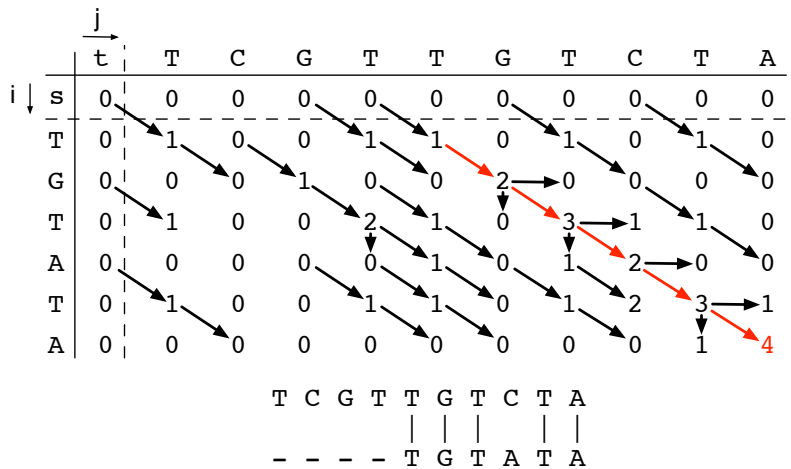


02 Alignment-Algorithmen

Smith & Waterman [Han06]

- Algorithmus zum lokalen Alignment
- Exaktes Verfahren (liefert immer optimales Alignment)
- Matrixfelder werden in Abhängigkeit voneinander berechnet
- Zeit- und Speicherkomplexität von $O(m \cdot n)$

$$M(i,j) = max \begin{cases} \underbrace{M(i-1,j) + g,}_{\text{Insertion}} \\ \underbrace{M(i,j-1) + g,}_{\text{Deletion}} \\ \underbrace{M(i-1,j-1) + p(s_i, t_j),}_{\text{Match / Mismatch}} \\ 0 \end{cases}$$

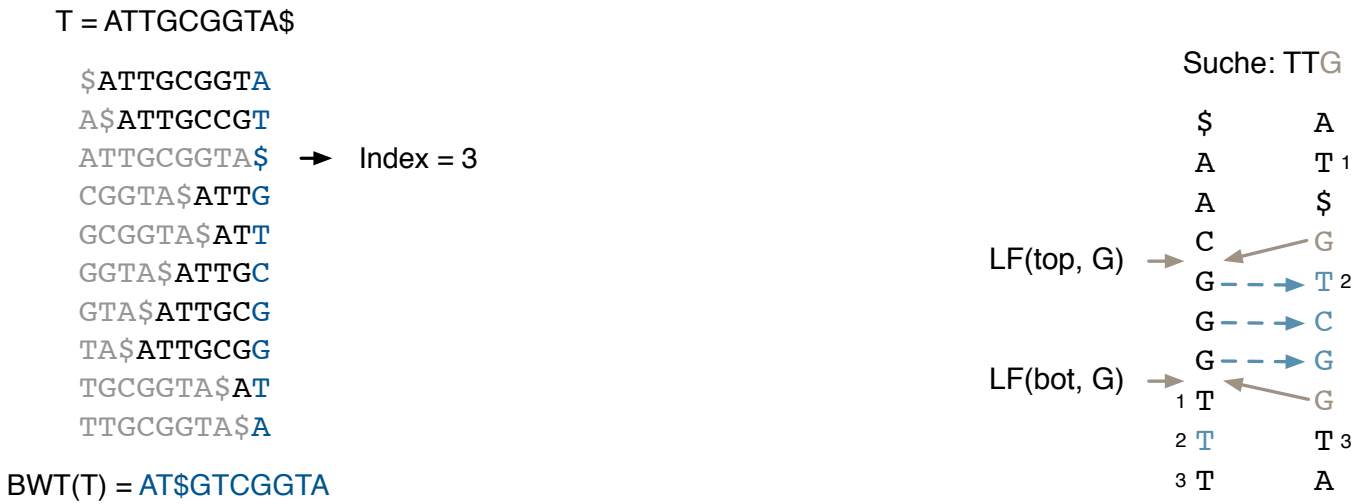


02 Alignment-Algorithmen

Short Read Mapper - Bowtie [TS09]

Gezielte Suche in Datenbanken mit Hilfe der Burrows-Wheeler-Transformation

- Hohe Geschwindigkeit, da nie die komplette Datenbank durchsucht wird
- Ausgelegt für Reads von 30 - 100 bp
- Einfügen von maximal 3 Mismatches über *Backtracking* (keine Gaps)



03 Auswahl
Auswahl des Algorithmus

1. Smith & Waterman
   - Kann durch seine einfache Struktur einfach auf SIMD-Plattformen umgesetzt werden
   - Hoher Aufwand, wenn die gesamte Matrix berechnet wird

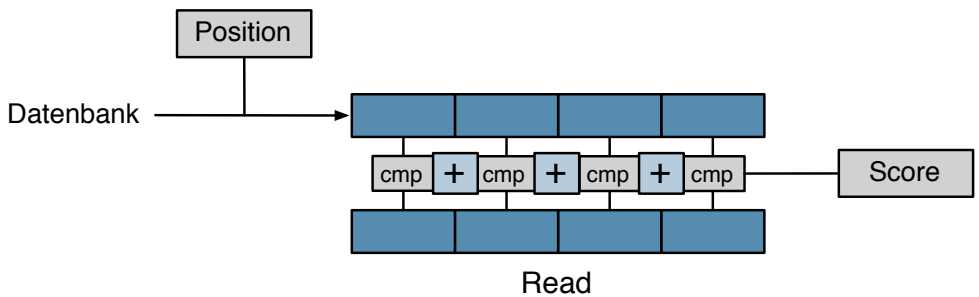
2. Bowtie
   - Hohe Geschwindigkeit
   - Eigentlicher Suchalgorithmus sehr schlecht parallelisierbar
   - Schlechtes Verhalten in Bezug auf Mismatches
   - Kein Einfügen von Gaps möglich

$\Rightarrow$ **Smith & Waterman-Algorithmus**

04 Implementierung - FPGA

Algorithmus

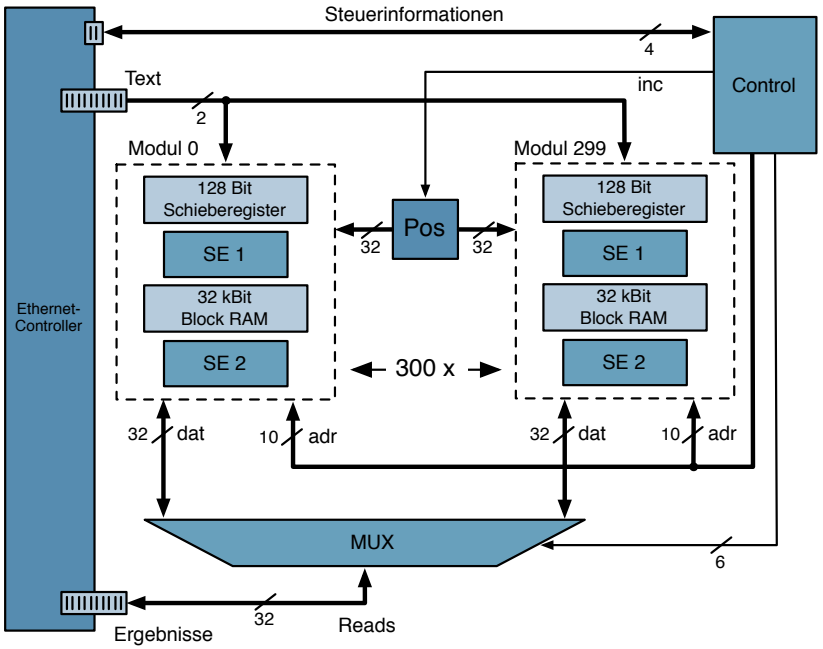
- Das Short Read Mapping Problem erfordert aufgrund der kurzen Sequenzen keine Gaps [Mar08]
- Mismatches sind aufgrund von Mutation oder Sequenzierungsfehlern notwendig
- Smith & Waterman-Algorithmus kann auf Diagonale reduziert werden



⇒ **Problem entspricht naiven Suchalgorithmus mit Mismatches**

04 Implementierung - FPGA

Gesamtsystem

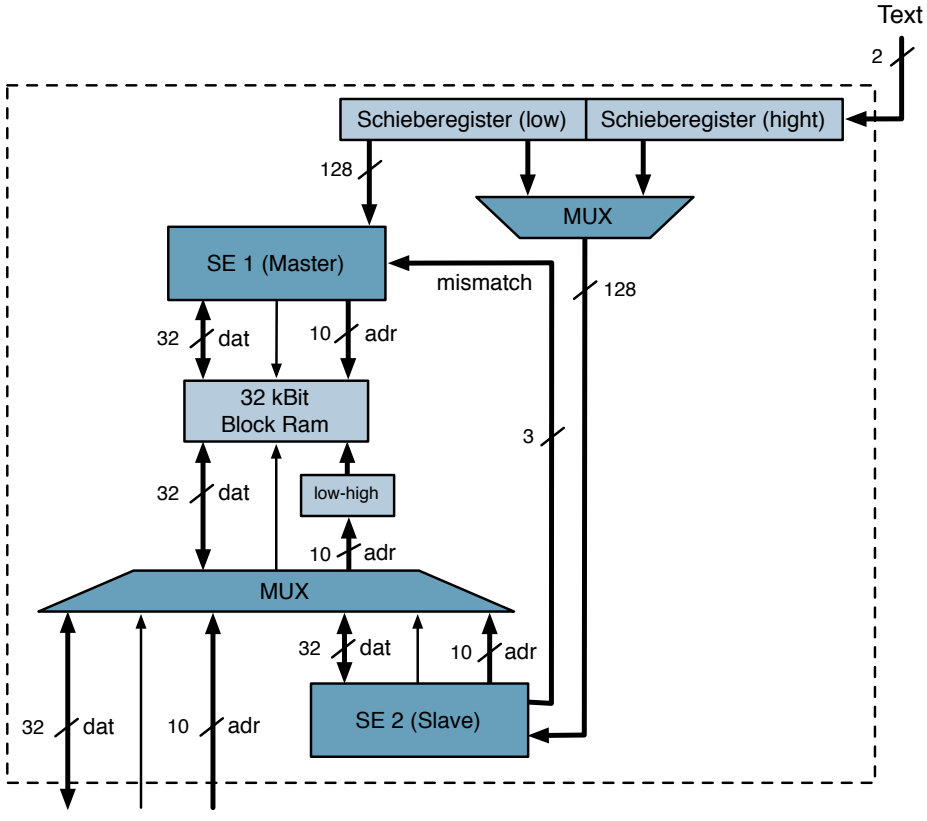


- 300 Module mit je zwei Sucheinheiten und einem BlockRAM
- Insgesamt $300 \times 2 = 600$ Sucheinheiten
- Jede Einheit bearbeitet einen Read mit 64 bp

04 Implementierung - FPGA

Sucheinheit

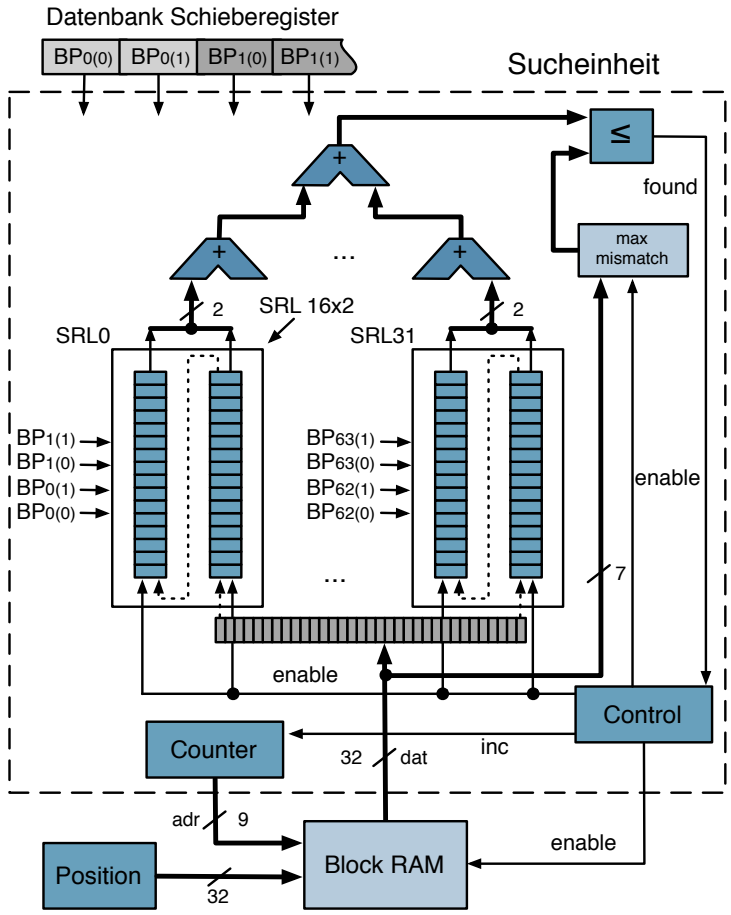
- Einheiten schreiben konfliktfrei durch eigenes Interface in den Block RAM
- Speichergröße wird dynamisch zwischen den beiden Reads verteilt
- Zwei Einheiten können zusammengeschaltet werden und Read mit 128 bp verarbeiten



04 Implementierung - FPGA

Sucheinheit

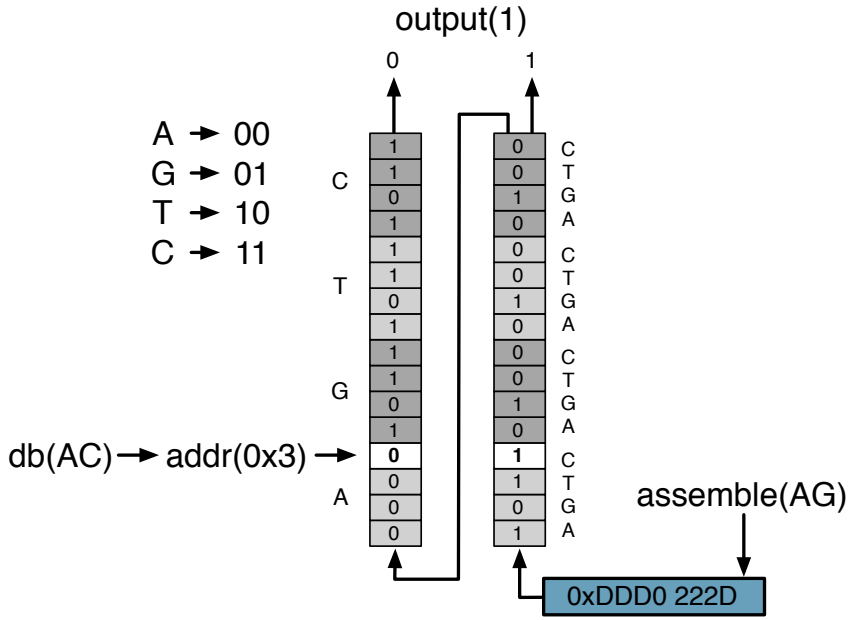
- Jeder SRL 16 × 2 [Xi10] enthält die Anzahl der Mismatches mit den beiden adressierenden Datenbank Basenpaaren
- 32 SRLs werden parallel bitweise geladen
- Baumförmiger sättigender 3-Bit Zähler (7 Mismatches)
- Schreiben der Position, wenn Mismatch-Grenze unterschritten wird
- Minimale Anzahl von Mismatches wird in Register gehalten



04 Implementierung - FPGA

Lesen der Reads

- Ein SRL 16 × 2 besteht aus zwei verketteten LUTs mit je sechs Eingängen
- Kodierung der Datenbank Basenpaare adressiert Speicherstelle
- Speicherstelle enthält Anzahl der Mismatches mit zwei Read Basenpaaren
- SRL-Kodierung der Reads per Software berechnet

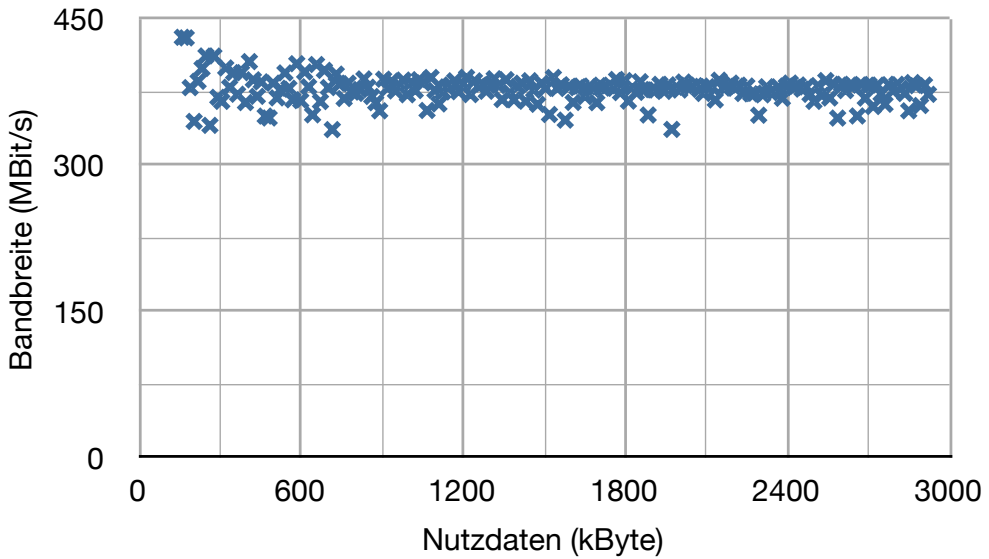


04 Implementierung - FPGA

Datenverbindung

Verbindung zwischen Host und FPGA realisiert über Gigabit-Ethernet (Layer 2)

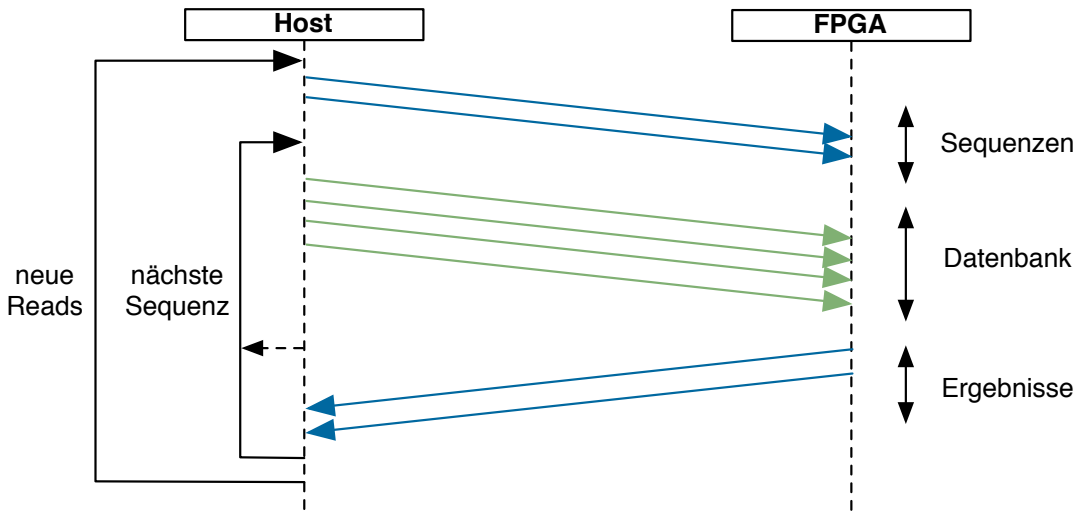
- Steuerung des FPGAs mit kurzen Paketen
- Übertragen der Reads und Ergebnisse
- Streaming der Datenbank über dynamische Flusskontrolle (∼ 400 MBit/s)
- Erkennen verlorener Pakete



04 Implementierung - FPGA

Host-Software I - Programmablauf

- Transformation der Datenbank von ASCII-Zeichen ins Binäre
- Ablaufsteuerung
  - Transformieren und Übertragen der Reads
  - Übertragen der Datenbank
  - Einsammeln und Aufbereiten der Ergebnisse



04 Implementierung - FPGA

Host-Software II - Kommandozeilenargumente

Parameter	Beschreibung
`-query <filename>`	Datei mit Reads im FASTA- oder FASTQ-Format
`-database <filename>`	Datenbank im FASTA-Format
`-bindb <filename>`	Bereits transformierte binäre Datenbank
`-output <filename>`	Ausgabe in Datei
`-transform`	Nur Transformation der ASCII-Datenbank
`-mismatch [int]`	Anzahl erlaubter Mismatches
`-positions`	Auflistung der Positionen der Reads
`-status`	Statusinformationen ausgeben
`-help`	Hilfetext anzeigen

04 Implementierung - FPGA

Hardwareressourcen

FPGA	Virtex-5 LX50T	Virtex-6 LX240T
Parallele Module	**50**	**300**
Parallele Sucheinheiten	**100**	**600**
Register	27.119 (94 %)	143.824 (47 %)
LUTs	23.136 (80 %)	121.601 (80 %)
Block-RAM	60 (100 %)	400 (96 %)
Taktrate	100 MHz	200 MHz
Vergleiche/s (Read mit 128 bp)[1]	5 Giga	60 Giga
Vergleiche/s (Read mit 64 bp)[1]	10 Giga	120 Giga
Erforderliche Datenrate	200 MBit/s	400 MBit/s

[1] Sucheinheiten × Taktrate

04 Implementierung - GPU

Algorithmus I

- Felder der Smith & Waterman-Matrix können diagonal berechnet werden
- Vereinfachungen gegenüber dem vollständigen Algorithmus:
  - Alle Scores über festgelegtem Schwellwert werden gesichert
  - Um Position ohne Backtracking zu berechnen zweite Matrix notwendig
- Parallelisierung einfach möglich

Datenbank

		N	N	N	A	G	A	T	A	C	N	N	N
	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	(0,0)	(0,0)	(0,0)	(1,1)	(0,0)	(1,3)	(0,0)	(1,5)	(0,0)	X	X	X
G	0	(0,0)	(0,0)	(0,0)	(0,0)	(2,1)	(0,0)	(1,3)	(0,0)	(1,5)	X	X	X
A	0	(0,0)	(0,0)	(0,0)	(1,1)	(0,1)	(3,1)	(2,1)	(2,3)	(0,3)	X	X	X
C	0	(0,0)	(0,0)	(0,0)	(0,0)	0,1	(2,1)	(3,1)	(1,1)	(3,3)	X	X	X

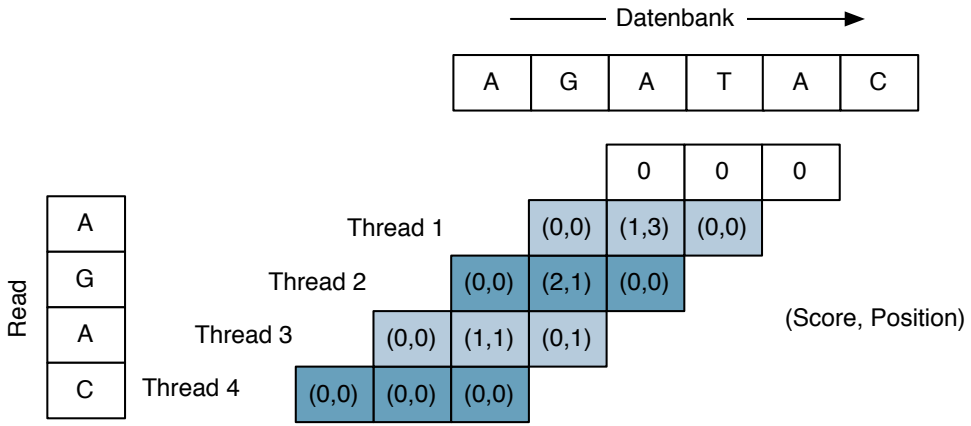
Read

(Score, Position)

04 Implementierung - GPU

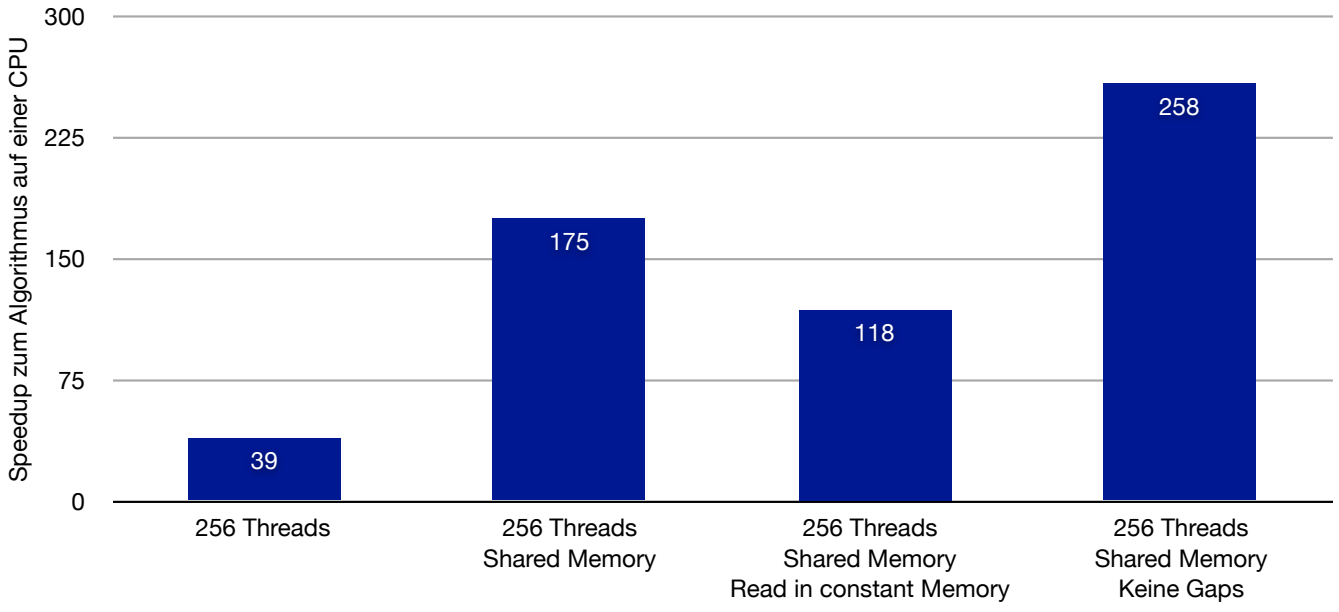
Algorithmus II

- Nur 3 Spalten breite Diagonale muss im Speicher gehalten werden
- Werden keine Gaps benötigt, sind weitere Vereinfachungen möglich:
  - Positionsmatrix nicht mehr notwendig
  - Keine Berechnungen des horizontalen- und vertikalen Gapscores erforderlich



04 Implementierung - GPU

Optimierung



Testsystem: Nvidia GeForce GTX 480

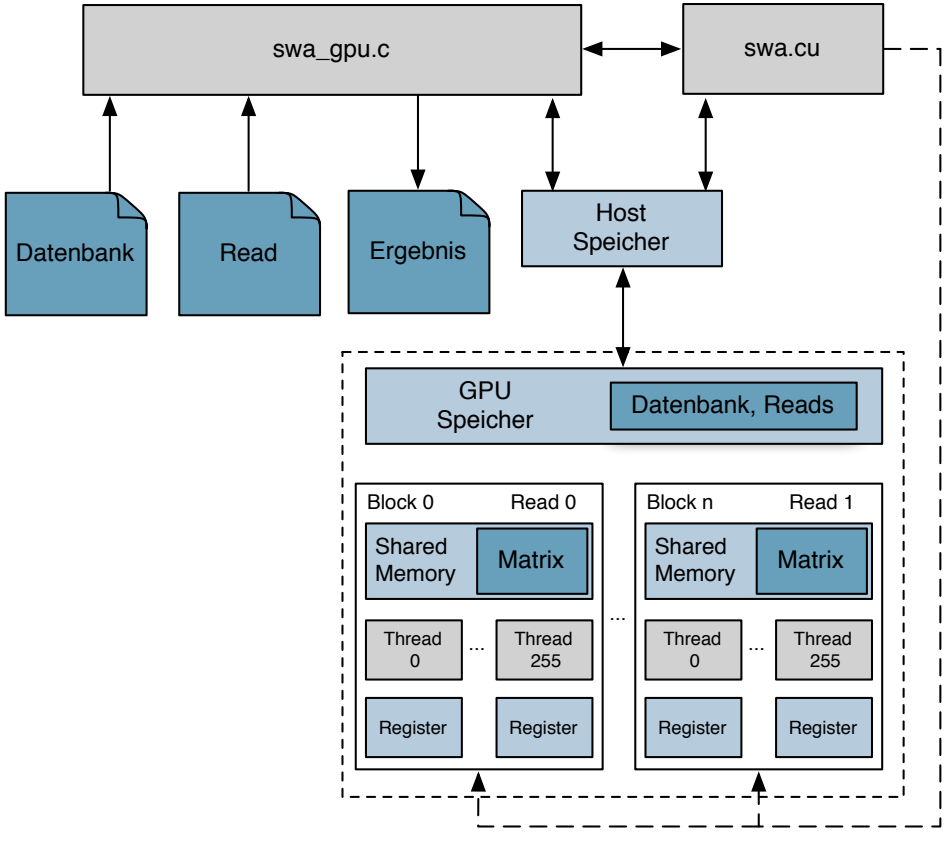
⇒ Optimale Laufzeit wird mit 256 Threads und Matrizen im Shared Memory erzielt

04 Implementierung - GPU

CUDA - Programm

Ablauf:

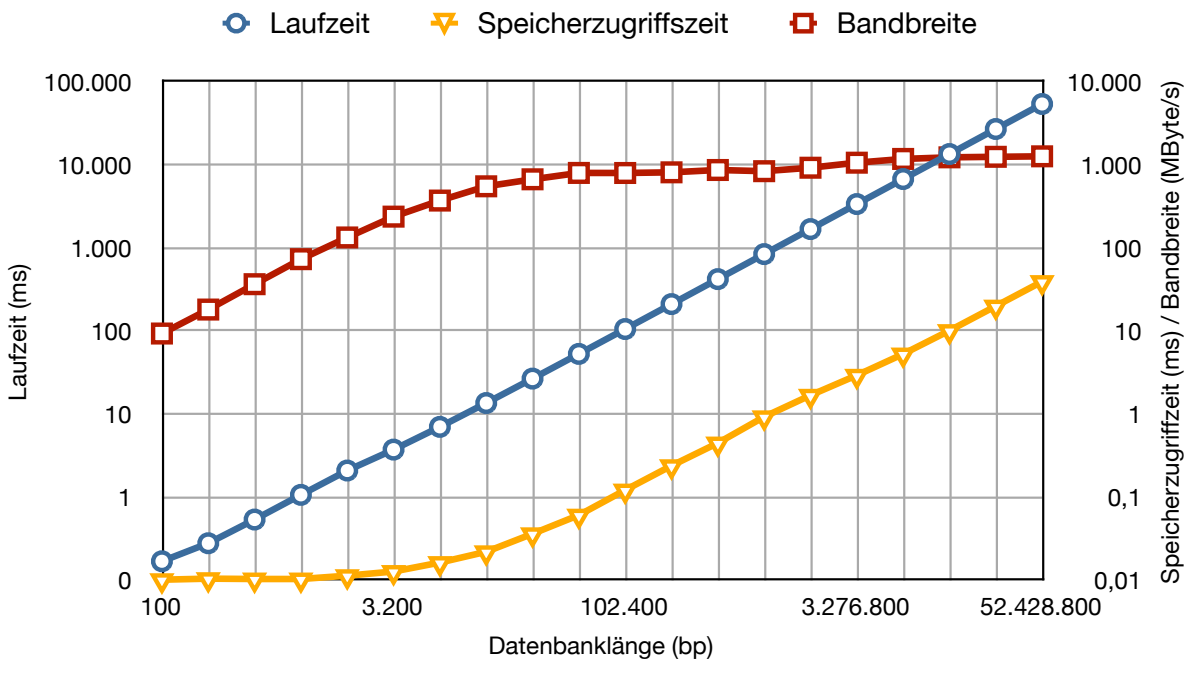
1. Übertragen von 1.000 Reads
2. Übertragen der ersten Datenbank-Sequenzen
3. Starten des Kernel mit 1.000 Blöcken und 256 Threads
4. Einsammeln der Ergebnisse



04 Implementierung - GPU

Leistung und Bandbreite

- Zeit zum Übertragen der Datenbank gering im Vergleich zur Suche
- Algorithmus skaliert mit Datenbankgröße und Anzahl der Reads



05 Bisherige Ergebnisse

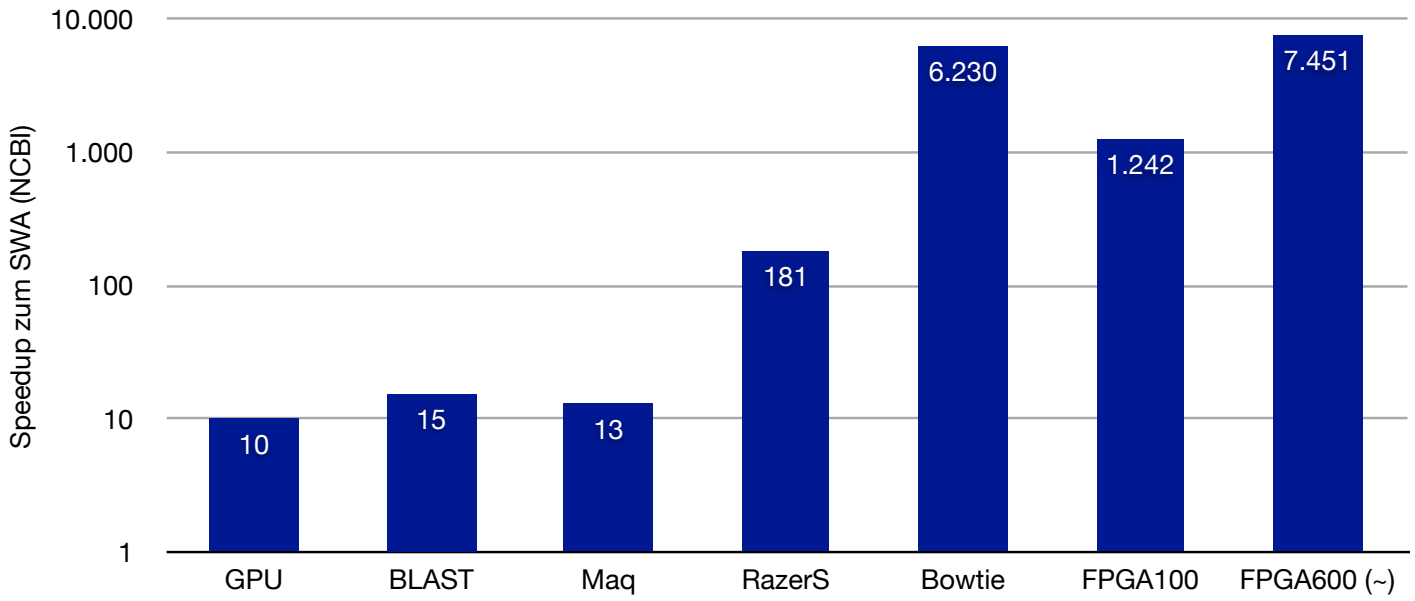
Laufzeit und Qualität

Programm	Laufzeit (s)	Reads mapped	Positionen
SWA (NCBI)	3.800,12	1.000 (100%)	> 5.000
SWA GPU	379,93	1.000 (100%)	3.823
BLAST (NCBI)	247,33	1.000 (100%)	> 5.000
Maq	293,34	777 (77%)	792
RazerS	21,02	869 (86%)	2.263
Bowtie	0,61	754 (75%)	1.515
FPGA100	3,06	1.000 (100%)	3.823
FPGA600	~ 0,51	1.000 (100%)	3.823

Datenbank: Homo sapiens chromosome 20 (7.281.792 bp)
Reads: 1.000 mit 50 bp und 0-3 mismatches in diskreter Gleichverteilung
Testsysteme: Intel Core 2 Duo 2,66 GHz, 3,8 GByte RAM
Nvidia GeForce GTX 480
Xilinx Virtex-6 XC6VLX240T

05 Bisherige Ergebnisse

Speedup



Datenbank: Homo sapiens chromosome 20 (7.281.792 bp)
Reads: 1.000 mit 50 bp und 0-3 mismatches in diskreter Gleichverteilung
Testsysteme: Intel Core 2 Duo 2,66 GHz, 3,8 GByte - Nvidia GeForce GTX 480 - Xilinx Virtex-6 XC6VLX240T

06 Zusammenfassung

Bisher erreicht:

- Auswahl eines Algorithmus
- Entwurf auf FPGA und GPU
- Einlesen des gängigen Formates für Reads und Datenbank (FASTA)
- Ausgabe in lesbares Format
- Datenübertragung mit voller Bandbreite ($\sim$ 400 MBit/s)
- Erste Vergleichsergebnisse

Weitere Aufgaben:

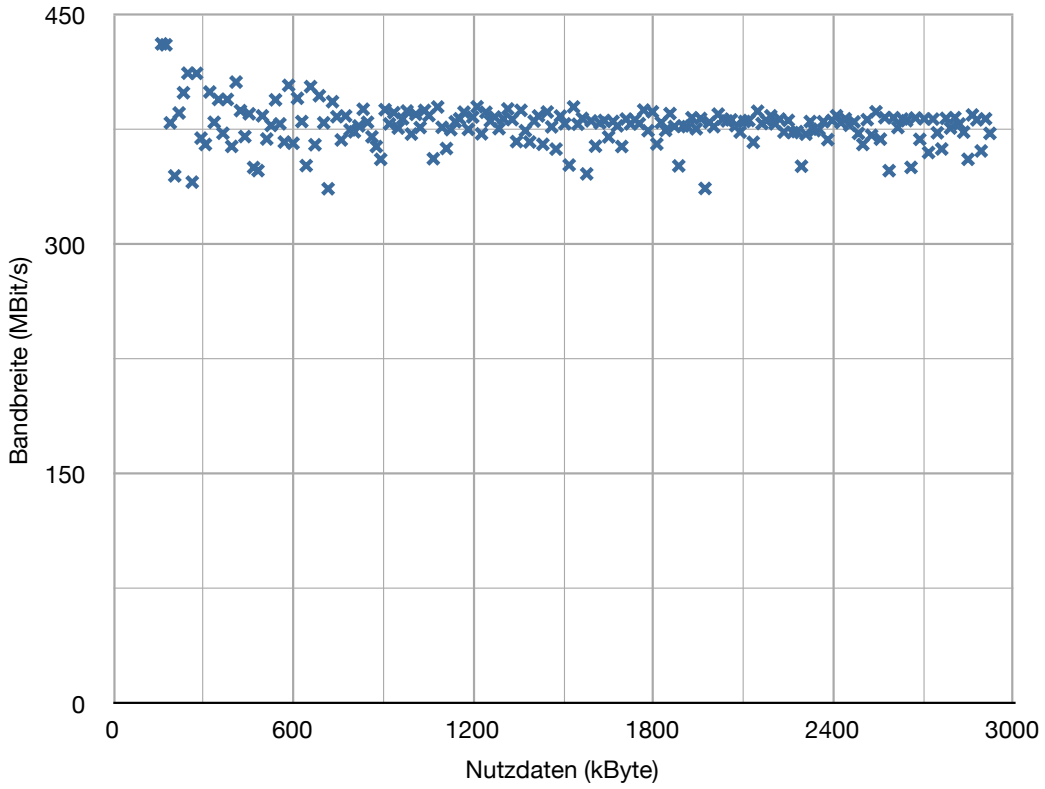
- Realisierung von 300 $\times$ 2 Einheiten
- Zusammenschalten von zwei Sucheinheiten
- Analyse und Softwareoptimierung (Positionen nachträglich berechnen)
- Ausführliche Tests und weitere Vergleichsergebnisse
- Als Option Ausgabe im SAM-Format [LHW09]

07 Quellen I

- [Han06] Hansen, Andrea
  Bioinformatik: Ein Leitfaden für Naturwissenschaftler 2006
- [TS09] Trapnell, C. and Salzberg, SL
  How to map billions of short reads onto genomes
  Nature biotechnology 2009
- [Mar08] Mardis, E.R.
  The impact of next-generation sequencing technology on genetics
  Trends in Genetics 2008
- [LHW09] Li, H. and Handsaker, B. and Wysoker, A. and Fennell, T. and Ruan, J.
  The sequence alignment/map format and SAMtools, 2009
- [Xi10] Xilinx
  Virtex-6 Libraries Guide for HDL Designs
  `http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_3/virtex6_hdl.pdf` 2010

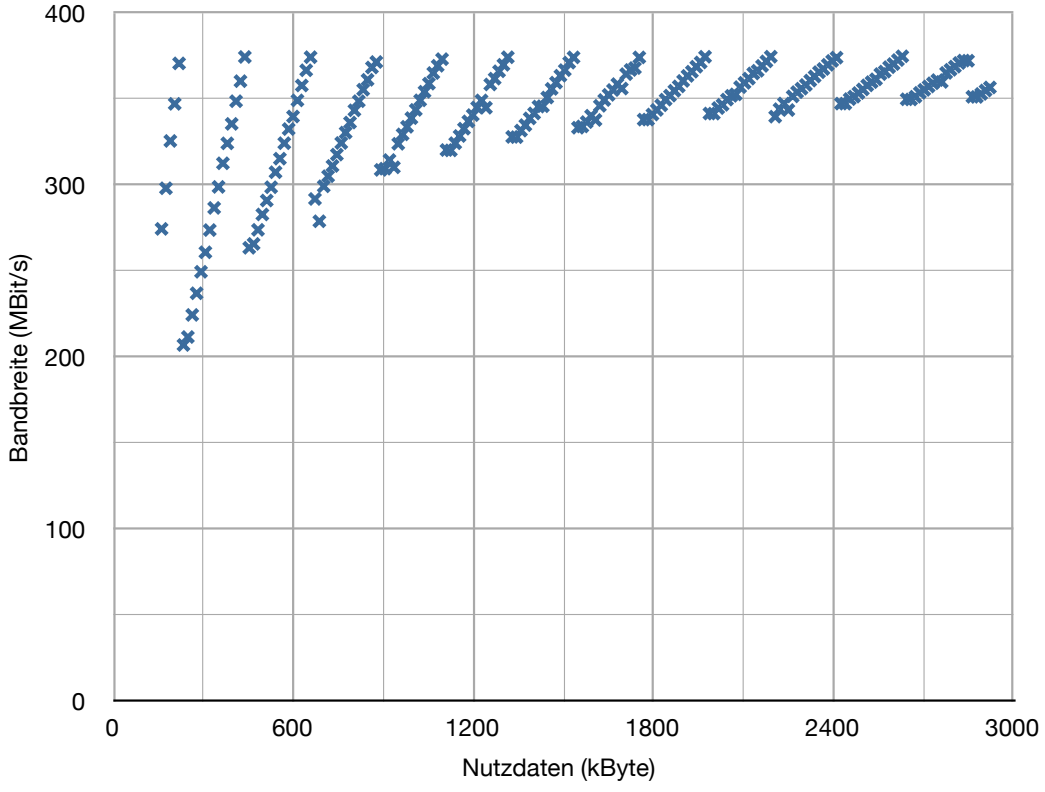
08 Anhang

Bandbreite - Virtex-6 (dynamische Flusskontrolle)



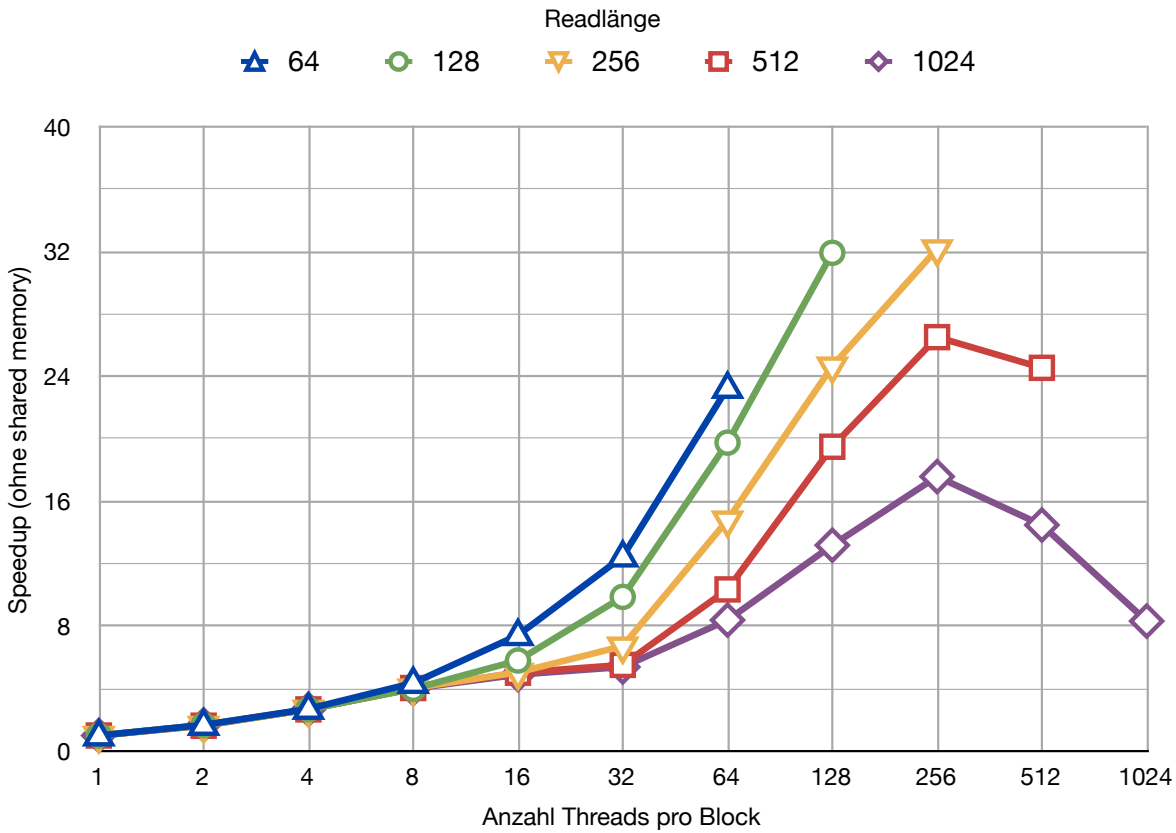
08 Anhang

Bandbreite - Virtex 6 (statische Flusskontrolle)



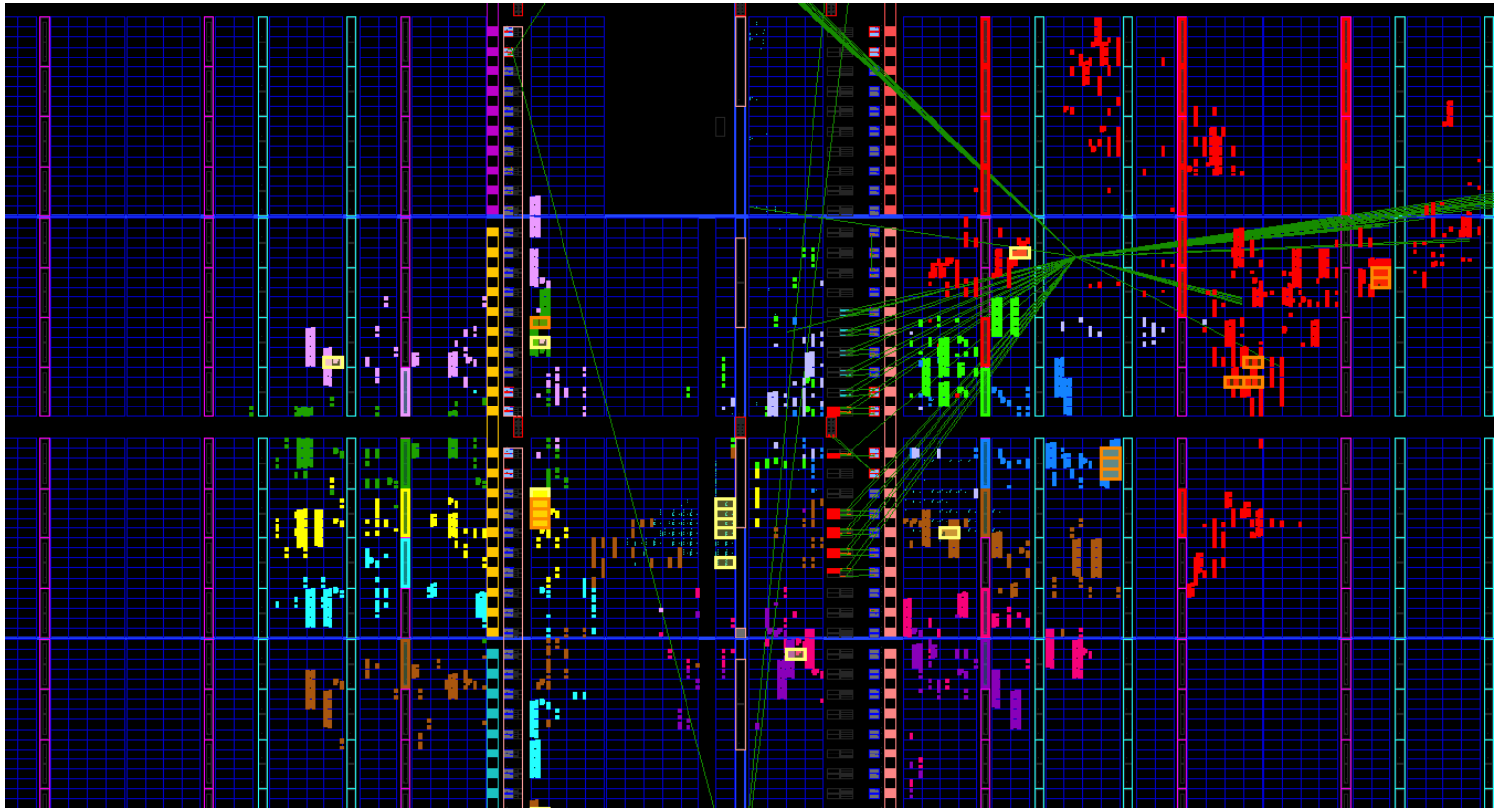
08 Anhang

Speedup für verschiedene Blockgrößen



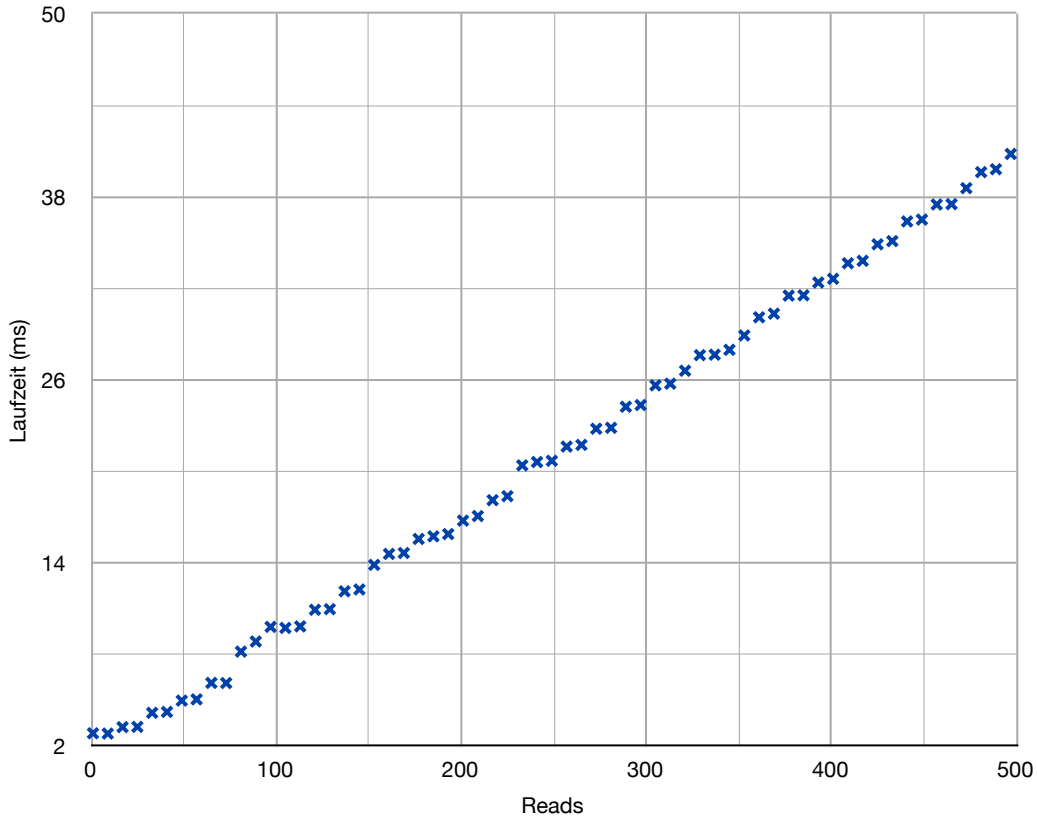
08 Anhang

Layout



08 Anhang

Skalierung in Abhängigkeit der Anzahl der Reads I



08 Anhang

Skalierung in Abhängigkeit der Anzahl der Reads II