TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

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

Implementierung des Genom-Alignments auf modernen hochparallelen Plattformen

Diplomverteidigung

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

Dresden, 28.06.2011

01 Einleitung

Problemstellung

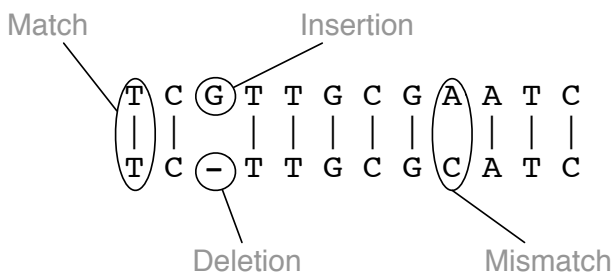
- Eine Aufgabe in der Molekularbiologie besteht darin, neu gewonnene Nukleotidsequenzen in Datenbanken bekannter Sequenzen zu suchen.
- Kostengünstige und hochparallele Sequenzierungsverfahren haben in den letzten Jahren die Anforderungen an die Suche verändert.
- Eine effiziente und kostengünstige Lösung des Problems auf einer parallelen Plattform ist notwendig und erfordert:
  - Eine Analyse der bisherigen Lösungsansätze,
  - Untersuchungen zur Parallelisierbarkeit auf FPGA und GPU,
  - Implementierung unter Berücksichtigung der typischer Nutzer,
  - Realisierung einer leistungsfähigen Datenkommunikation (FPGA) und
  - Funktionsnachweis und Vergleich mit leistungsfähigen Programmen.

02 Grundlagen

Klassisches Alignment-Problem

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

- Nukleotidesequenzen bestehen aus den Zeichen $\Sigma = \{A, G, T(\equiv U), C\}$.
- Um einen hohen Score zu erreichen, werden *Mismatches* und *Gaps* eingefügt.
- Wichtige Anwendung ist die inexakte Suche in Datenbanken, um Eigenschaften der Sequenzen zu ermitteln:
  - Die durch das Sanger-Verfahren generierten Sequenzen haben eine Länge von 800 bis 1.000 Basenpaaren.
  - Ergebnis einer Datenbanksuche ist die Position, an der ein bestimmter Schwellwert für den Score überschritten wird.

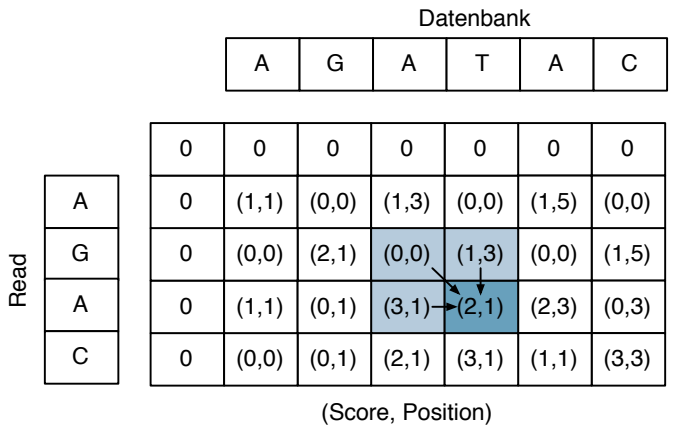


02 Grundlagen

Smith & Waterman-Algorithmus

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

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

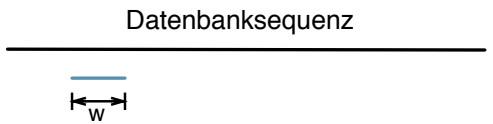


02 Grundlagen

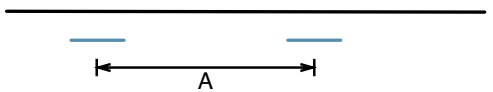
Basic Local Alignment Search Tool

Heuristisches Verfahren zur Datenbanksuche für lange Sequenzen (bis zu 1.000 Basenpaare) bestehend aus drei Teilschritten:

1. Exakter Treffer der Länge *w* wird gesucht.



2. Suche eines zweiten Hits mit maximalem Abstandes *A*.



3. Hits werden mit Matches, Mismatches und Gaps ausgedehnt.

02 Grundlagen

Short-Read Mapping Problem [TS09]

- Mit der Sanger-Sequenzierung wurden wenige lange Sequenzen geliefert.
- Technologische Weiterentwicklungen haben zu automatisierten hochparallelen Sequenzierern geführt (Next-Generation Sequencing) [RF09]:
  - Sequenzen (*Reads*) haben nur eine Länge von wenigen Basenpaaren.
  - Ein Durchlauf erzeugt mehrere Millionen Reads in kurzer Zeit.
- Score verliert an Bedeutung und wird auf Mismatches oder *Hamming-Distanz* reduziert.

Methode	Readlänge (bp)	Reads pro Durchlauf	Durchlaufzeit
Sanger-Verfahren	800	1	10 Stunden
ABI SOLiD	$\sim$ 50	85.000.000	6 Tage
Helicos Heliscope	30 - 38	800.000.000	8 Tage
Illumina Genome Analyzer	36 - 175	40.000.000	3 - 6 Tage

$\Rightarrow$ **Neue Anforderungen an die Programme.**

02 Grundlagen

Short-Read Mapper

Programme, die an die Anforderungen kurzer Reads angepasst sind und große Datenmengen verarbeiten können, werden als *Short-Read Mapper* bezeichnet.

1. **Burrows-Wheeler Transformation** - Bowtie und SOAP2
   - Suche der Reads mit dem FM-Index Algorithmus [FMMN04].
   - Hohe Geschwindigkeit, da nie die gesamte Datenbank durchsucht wird.
   - Einfügen von maximal drei Mismatches über Backtracking.
2. **Spaced-Seed Indexing** - Maq und PASS
   - Unterteilung der Reads in Bereiche fester Länge.
   - Suche der kurzen Bereiche in der Datenbank.
3. **Q-Gram Counting** - RazerS
   - Berechnung der Anzahl kurzer Teilsequenzen für einen Treffer über das Q-Gram Lemma [BCF+99].
   - Durch Länge der Teilsequenzen von wenigen Basenpaaren ist eine schnelle Suche mit Mismatches und Gaps möglich.

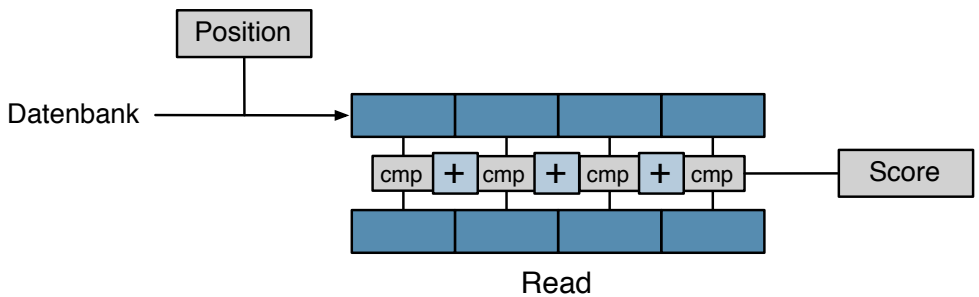
03 Untersuchungen zur Parallelisierbarkeit

1. **Modifizierter Smith & Waterman-Algorithmus**
   - Exakte Ergebnisse und einfache Anpassung an das Short-Read Mapping.
   - Einfache parallele Berechnung der Matrixelemente und gleichmäßige Datenbankzugriffe.
2. **Spaced Seed Indexing**
   - Die Suche mit Seeds fester Größe lässt sich auf dem FPGA leicht realisieren.
3. **Burrows-Wheeler Transformation**
   - Speicherzugriffe in der Datenbank nicht vorhersagbar.
   - Ein Zugriff mehrerer Einheiten auf einen Speicher wird zum Engpass.
4. **Q-Gram Counting**
   - Umfangreiche Nachbearbeitung der markierten Positionen auf dem Host-System nötig.
   - Parallelisierung möglich, aber komplexe Umsetzung in Hardware.

04 Implementierung - FPGA

Algorithmus

- Das Short-Read Mapping erfordert aufgrund der kurzen Sequenzen keine Gaps [TS09].
- Mismatches sind aufgrund von Mutation oder Sequenzierungsfehlern notwendig.

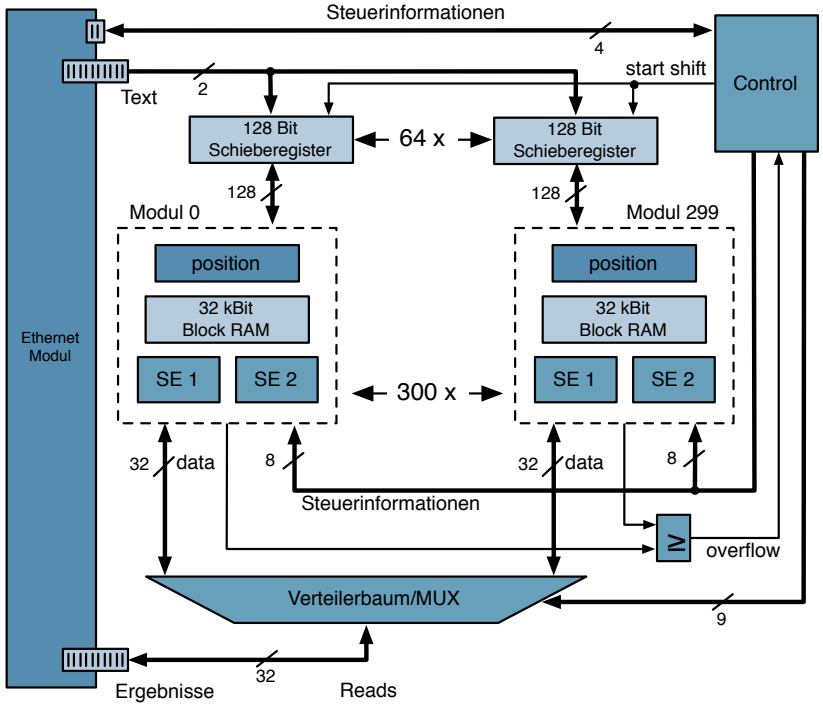


⇒ **Problem entspricht dem naiven Suchalgorithmus mit Mismatches.**

04 Implementierung - FPGA

Gesamtsystem

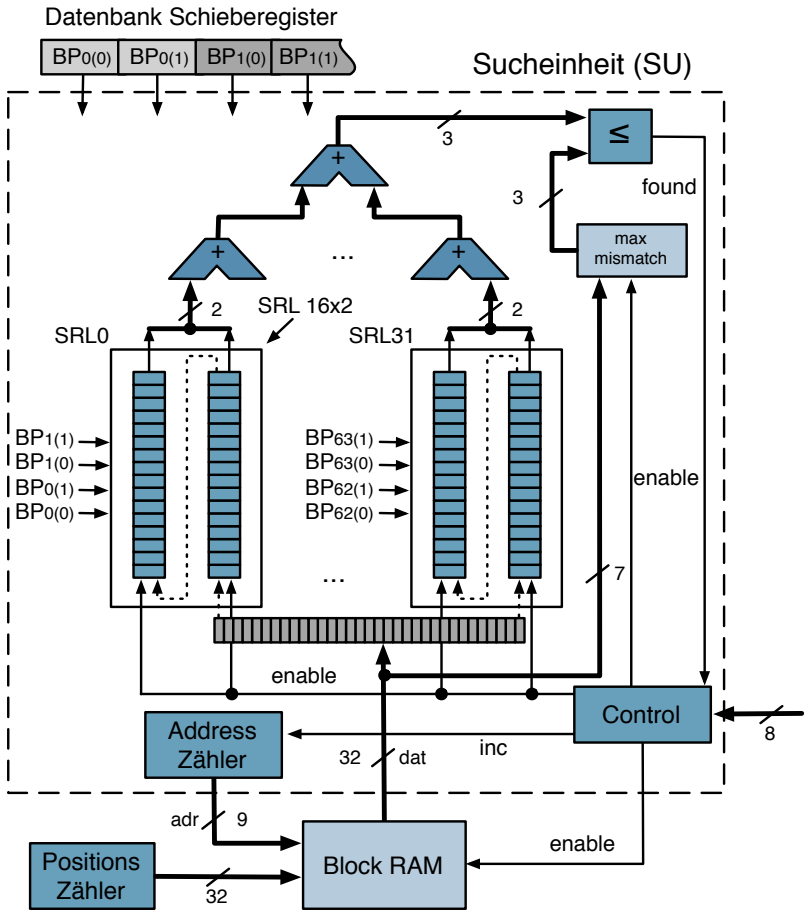
- 300 Module mit je zwei Einheiten zur Suche und einem BlockRAM (Xilinx Virtex-6 XC6VLX240T).
- Insgesamt $300 \times 2 = 600$ Sucheinheiten.
- Jede Einheit bearbeitet einen Read mit 64 Basenpaaren.
- Bei einer Taktrate von 200 MHz wird eine Leistung von 120 Giga Vergleiche/s erreicht.



04 Implementierung - FPGA

Sucheinheit

- Jeder SRL 16 × 2 [Xil10] enthält die Anzahl der Mismatches mit den beiden adressierenden Basenpaaren der Datenbank.
- 32 SRLs werden parallel bitweise geladen.
- Baumförmiger sättigender 3-Bit Zähler (7 Mismatches).
- Schreiben der Position, wenn Mismatch-Schwellwert unterschritten wird.



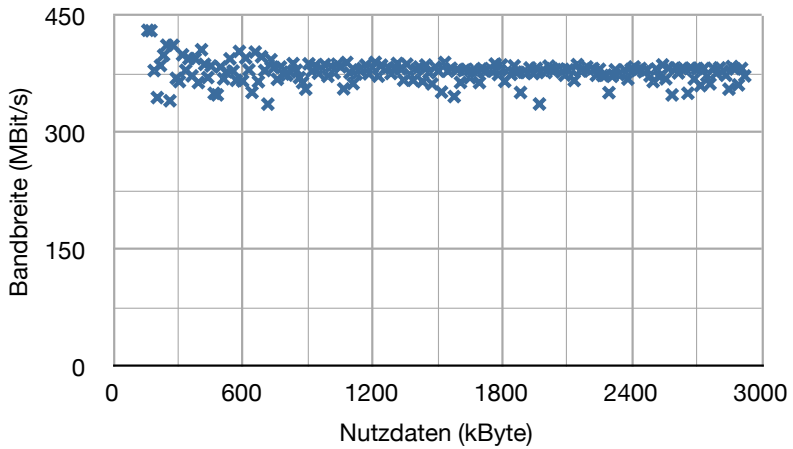
04 Implementierung - FPGA

Datenverbindung

Verbindung zwischen Host und FPGA realisiert über Gigabit-Ethernet auf Ebene der MAC-Adressen (Sicherungsschicht - Layer 2) [BH01].

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

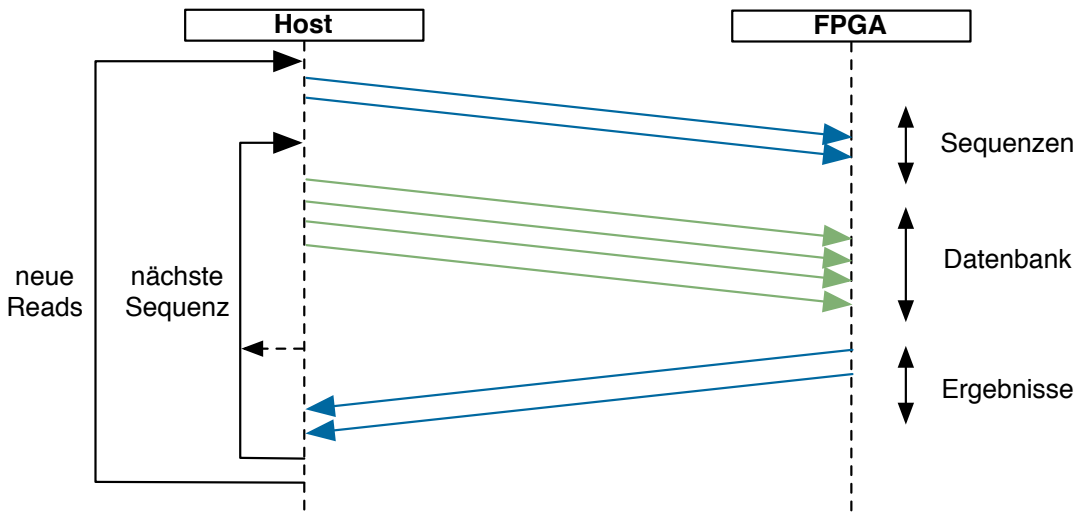
$\Rightarrow$Teilweise Realisierung eines TCP-ähnlichen Protokolls (Transportschicht - Layer 4).



04 Implementierung - FPGA

Host-Software I - Programmablauf

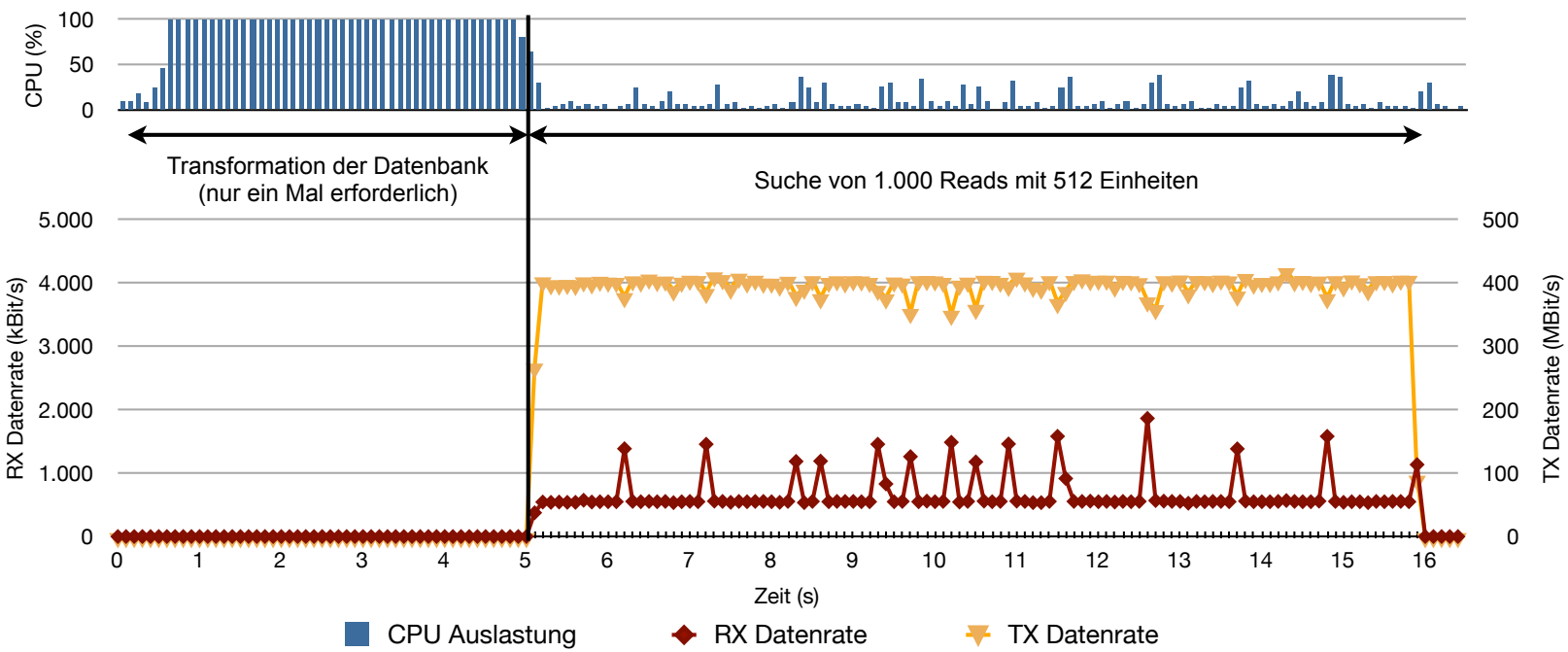
- Transformation der Datenbank von ASCII-Zeichen ins Binäre
- Ablaufsteuerung:
  - Transformieren und Übertragen der Reads,
  - Übertragen der Datenbank,
  - Einsammeln und Aufbereiten der Ergebnisse.



04 Implementierung - FPGA

Host-Software II - Nutzbarkeit

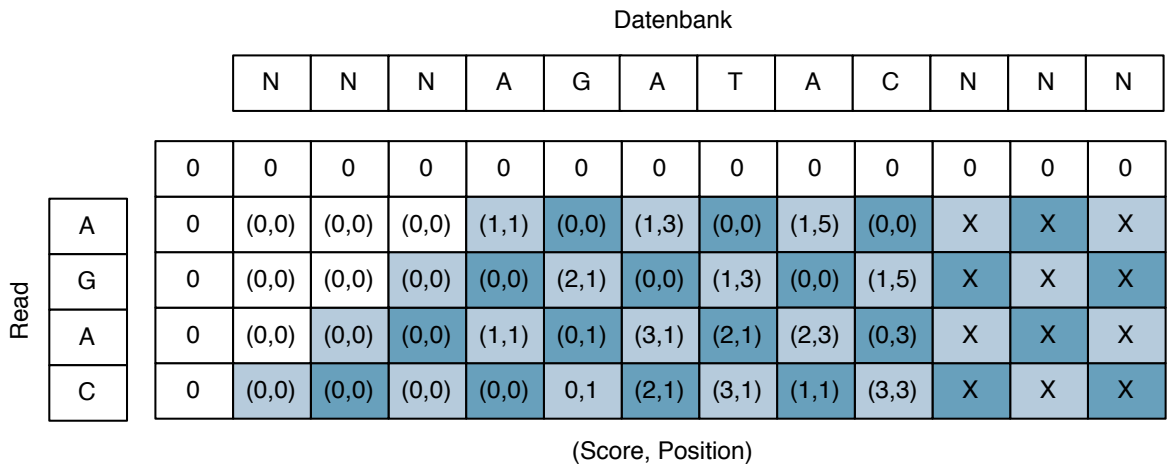
- Ausführung des Programms über Kommandozeile.
- Unterstützung der gebräuchlichen Eingabeformate (FASTA, FASTQ).
- Optionale Ausgabe im SAM-Format [LHW+09].



04 Implementierung - GPU

Algorithmus I

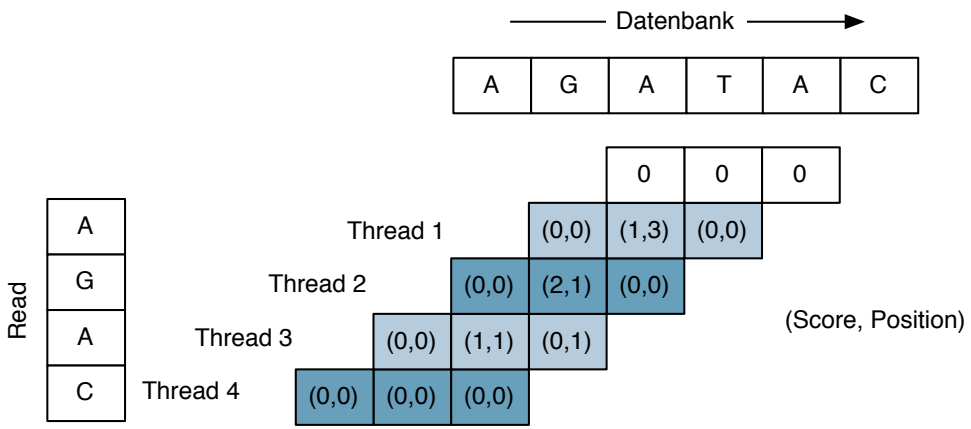
- Felder der Smith & Waterman-Matrix können diagonal berechnet werden.
- Vereinfachungen gegenüber dem vollständigen Algorithmus:
  - Alle Scores über festgelegtem Schwellwert werden gesichert.
  - Um Position ohne Backtracking zu berechnen, ist eine zweite Matrix notwendig.



04 Implementierung - GPU

Algorithmus II

- Nur eine 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.



05 Ergebnisse

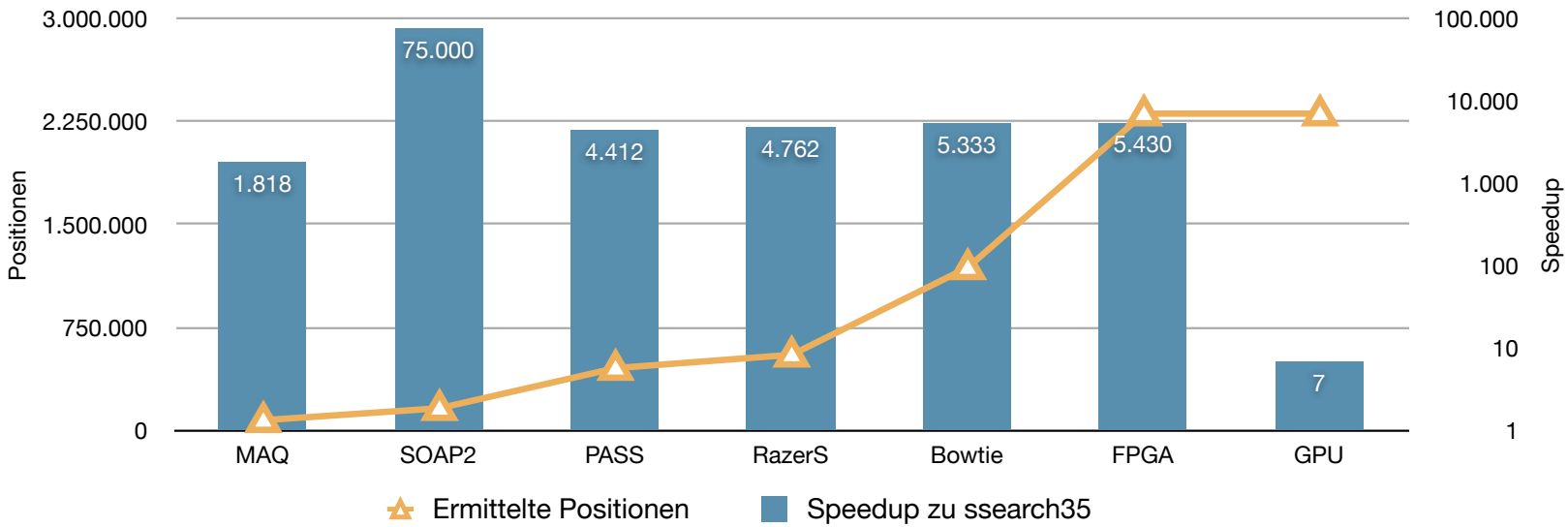
Laufzeit und Qualität

Programm	Datenbank-Transformation (h:min:sec)	**Reads: 100.000 × 38 bp Maximal 2 Mismatches**			**Reads: 100.000 × 50 bp Maximal 3 Mismatches**		
		Laufzeit (h:min:sec)	Mapped (%)	Positionen	Laufzeit (h:min:sec)	Mapped (%)	Positionen
ssearch	—	305:23:00	100	—	333:20:00	100	—
Maq	00:00:08	00:03:26	100	100.000	00:11:03	76	76.874
SOAP2	00:04:21	00:00:12	100	476.991	00:00:16	76	163.913
PASS	—	00:03:32	99	601.683	00:04:32	99	455.862
RazerS	—	00:03:38	100	707.067	00:04:12	100	552.573
Bowtie	00:07:48	00:00:57	85	1.967.427	00:03:45	77	1.188.046
FPGA	00:00:05	00:03:46	100	5.475.284	00:03:41	100	2.306.253
GPU	—	43:39:00	100	5.475.284	47:46:00	100	2.306.253

Datenbank: Homo sapiens chromosome 1 (222.388.987 bp)
Reads: Mismatches in diskreter Gleichverteilung
Testsysteme: Intel Core 2 Duo 2,66 GHz, 3,8 GByte - Nvidia GeForce GTX 480 - Xilinx Virtex-6 XC6VLX240T

05 Ergebnisse

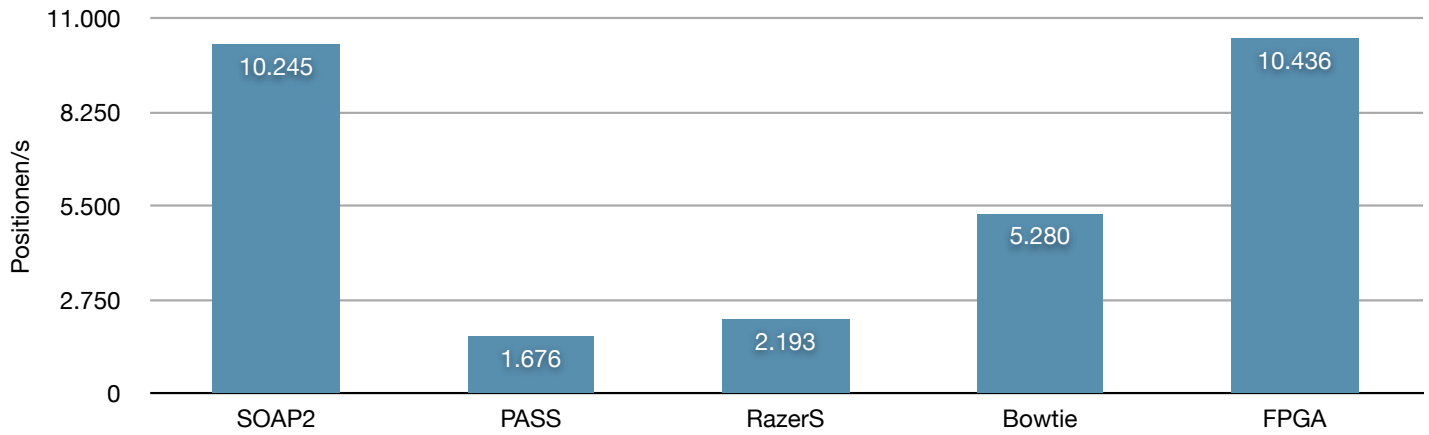
Speedup zu ssearch und Ergebnisqualität



Datenbank: Homo sapiens chromosome 1 (222.388.987 bp)
Reads: 100.000 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

05 Ergebnisse

Performance



Datenbank: Homo sapiens chromosome 1 (222.388.987 bp)
Reads: 100.000 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 und Ausblick

- Zusammenfassung:
  - Aufgrund der Weiterentwicklungen in den Sequenzierungsverfahren sind neue Programme erforderlich.
  - Ein naiver Suchalgorithmus kann einfach auf parallele Plattformen übertragen werden.
  - Die FPGA-Implementierung erreicht ähnlich hohe Geschwindigkeit wie heuristische leistungsfähige Programme, der GPU-Ansatz ist deutlich langsamer.
  - Aufgrund der geringen Auslastung des Host-Systems ist eine Integration in ein bestehendes System möglich.
- Ausblick:
  - Weitere Optimierung der FPGA-Implementierung.
  - Direkte Anpassung der Hardware an die Länge der Reads.
  - Für längere Reads Erweiterung auf Gaps durch vollständigen Smith & Waterman-Algorithmus oder Q-Gram Counting.

07 Quellen I


BURKHARDT, S. ; CRAUSER, A. ; FERRAGINA, P. ; LENHOF, H.P. ; RIVALS, E. ; VINGRON, M.:
q-gram based database searching using a suffix array (QUASAR).
In: *Proceedings of the third annual international conference on Computational molecular biology* ACM, 1999, S. 77–83


BADACH, A. ; HOFFMANN, E.:
Technik der IP-Netze.
Hanser, 2001


FERRAGINA, P. ; MANZINI, G. ; M
ÄKINEN, V. ; NAVARRO, G.:
An alphabet-friendly FM-index.
In: *String Processing and Information Retrieval* Springer, 2004, S. 228–228


LI, H. ; HANDSAKER, B. ; WYSOKER, A. ; FENNELL, T. ; RUAN, J. ; HOMER, N. ; MARTH, G. ; ABECASIS, G. ; DURBIN, R.:
The sequence alignment/map format and SAMtools.
In: *Bioinformatics* 25 (2009), Nr. 16, S. 2078

07 Quellen II

REIS-FILHO, J.:
Next-generation sequencing.
In: *Breast Cancer Research* 11 (2009), Nr. Suppl 3, S. S12

TRAPNELL, C. ; SALZBERG, SL:
How to map billions of short reads onto genomes.
In: *Nature biotechnology* 27 (2009), Nr. 5, S. 455

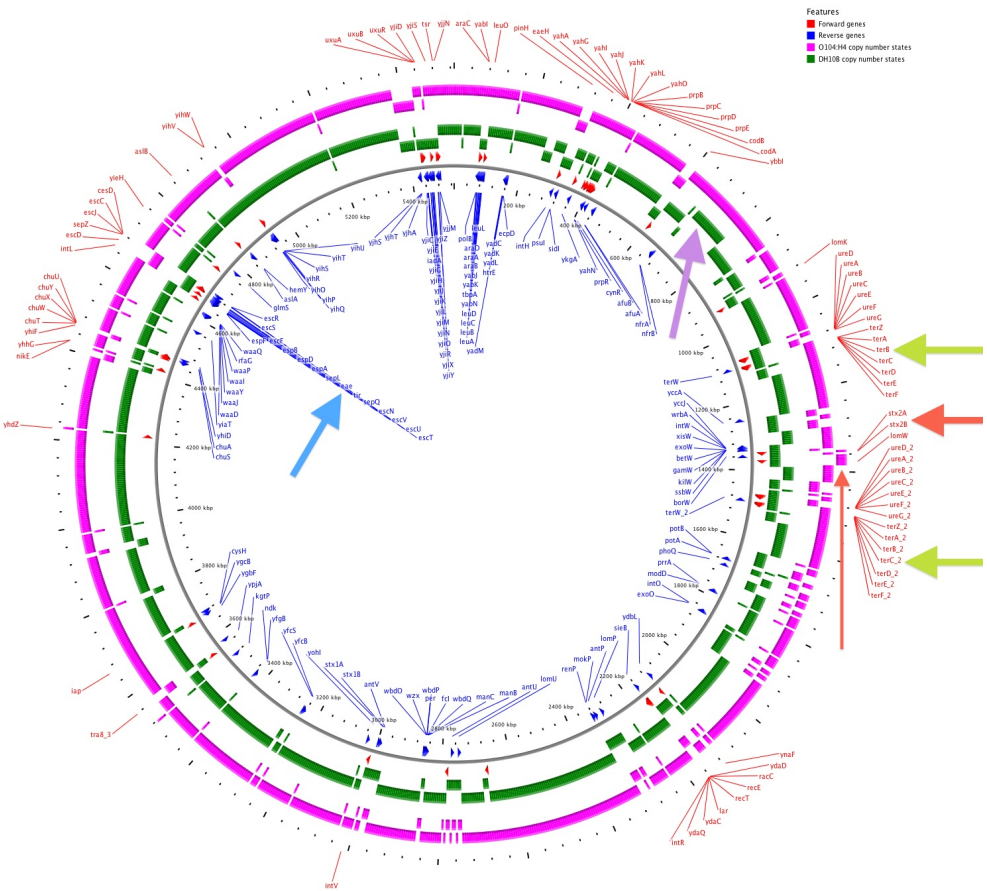
XILINX:
Virtex-6 Libraries Guide for HDL Designs.
`http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_3/virtex6_hdl.pdf`, September 2010

08 Anhang - Grundlagen

Next-Generation Sequencing - Beispiel: EHEC

- Sequenzierung mit Geräten von Ion Torrent und Illumina.
- Ergebnis: Hybrid-Klon bekannter Stämme.

https://github.com/ehec-outbreak-crowdsourced/



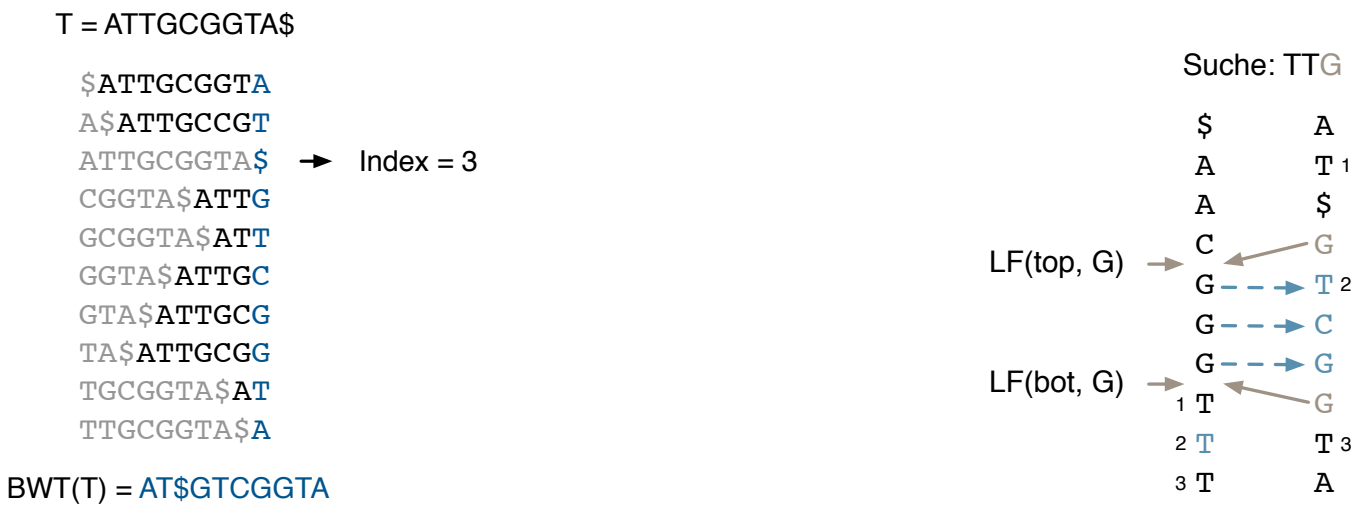
Escherichia coli O157:H7 EDL933, complete genome. - 1..5528445

08 Anhang - Grundlagen

Short Read Mapper - Bowtie

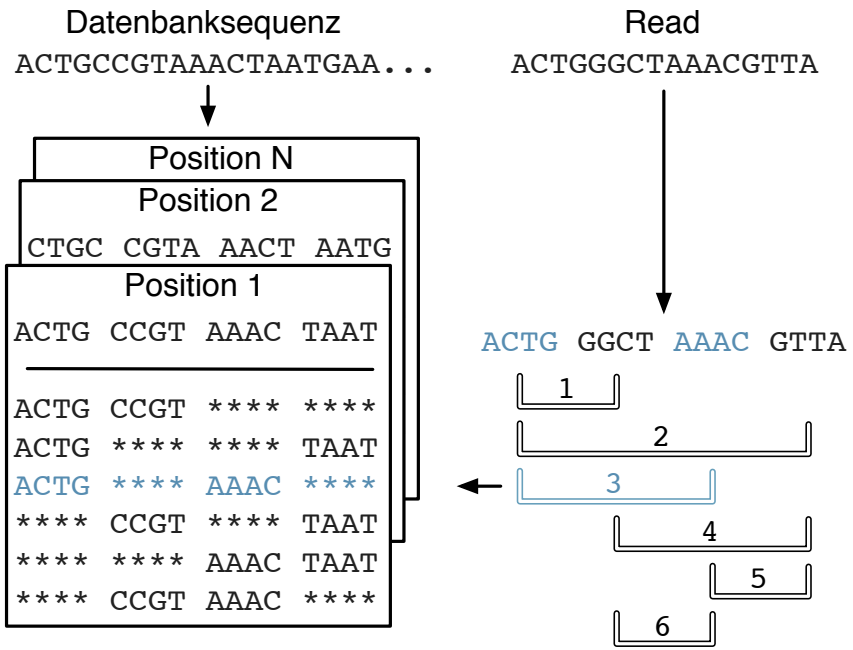
Gezielte Suche in Datenbanken mit Hilfe der Burrows-Wheeler-Transformation.

- 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).



08 Anhang - Grundlagen

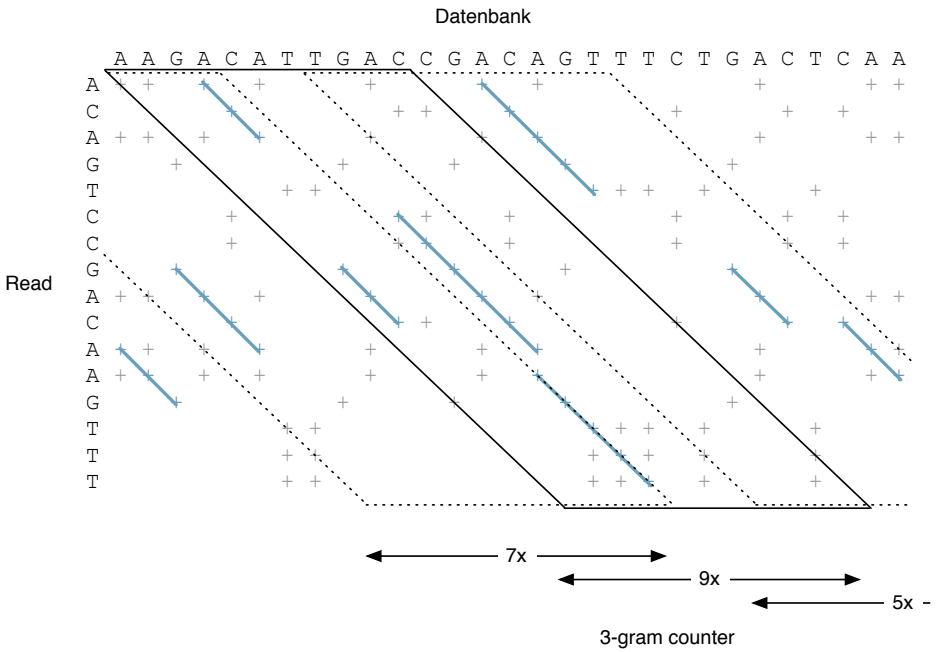
Short Read Mapper - Maq



08 Anhang - Grundlagen

Short Read Mapper - RazerS

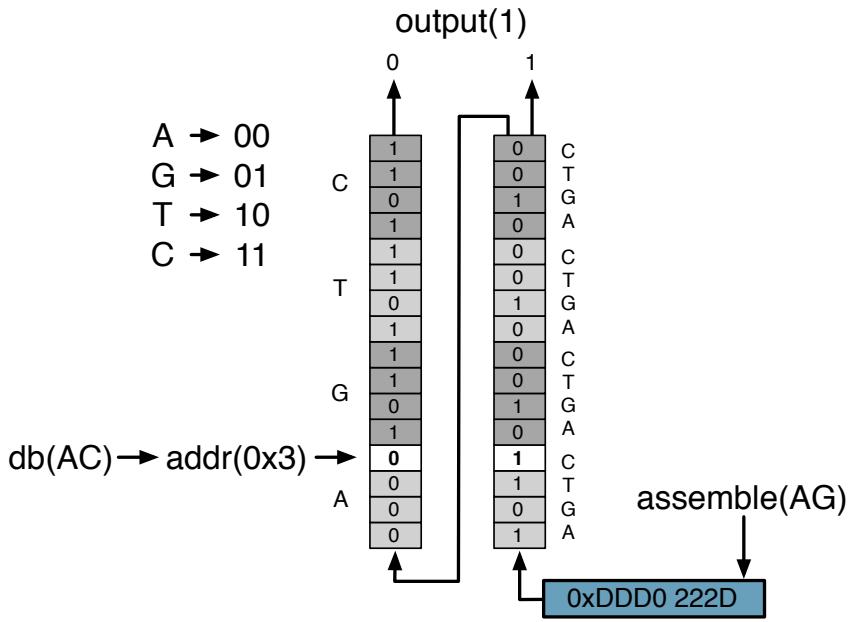
$$t = n + 1 - (k + 1) \cdot q \tag{1}$$



08 Anhang - FPGA

Lesen der Reads

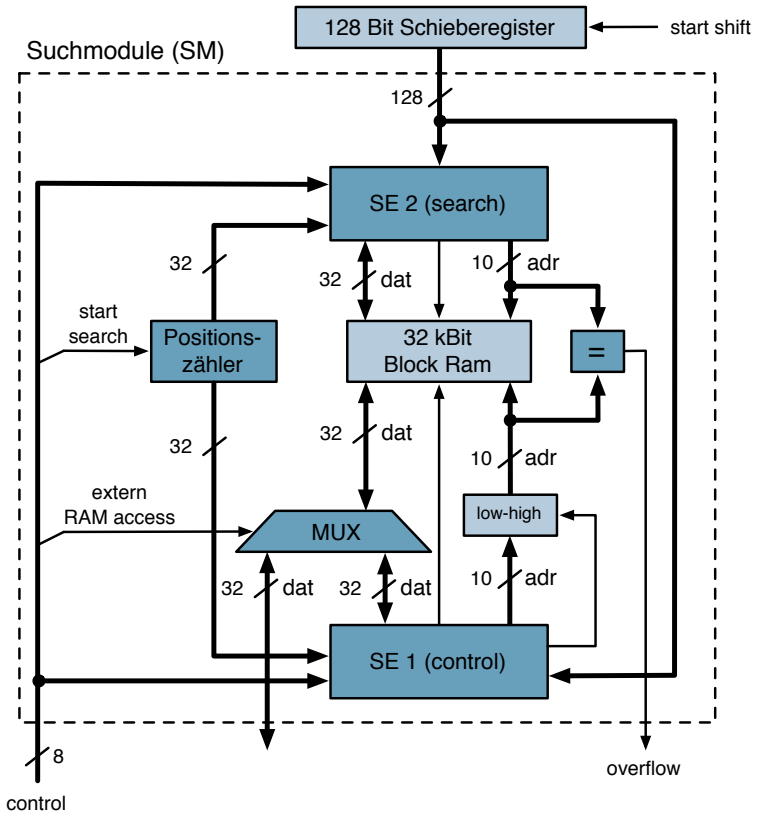
- Ein SRL 16 × 2 besteht aus zwei verketteten LUTs mit je sechs Eingängen.
- Kodierung der Datenbank Basenpaare adressiert Speicherstelle.
- Speicherstelle enthält Anzahl der Mismatches mit zwei Read Basenpaaren.
- SRL-Kodierung der Reads per Software berechnet.



08 Anhang - FPGA

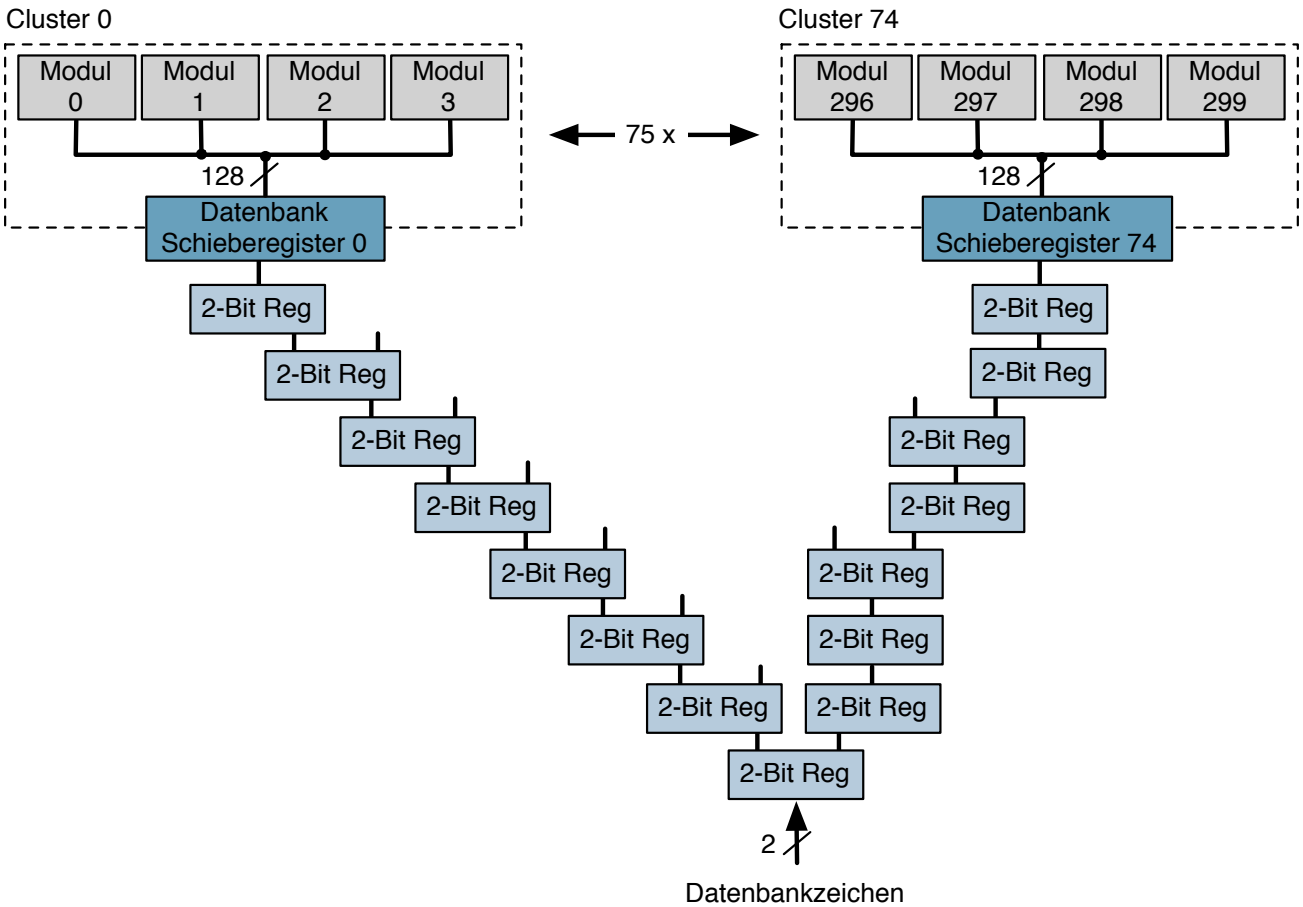
Sucheinheit

- 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.



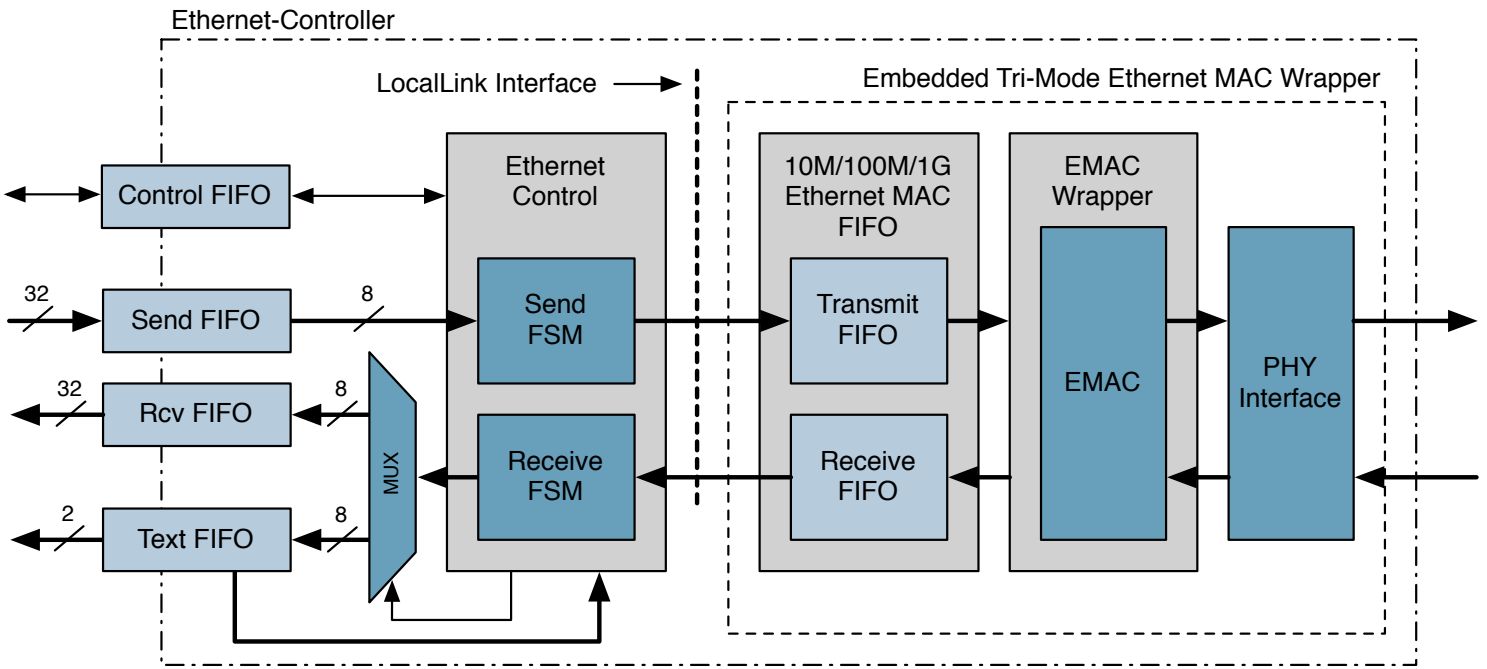
08 Anhang

Verteilung der Einheiten



08 Anhang

Aufbau des Ethernet-Modules



08 Anhang - FPGA

Hardwareressourcen

Modul	Suchmodul	Ethernet	600 Einheiten
Register	165	1.161	76.768 (25 %)
LUTs	405	964	127.921 (85 %)
Slices	162	400	35.362 (93 %)
Unbenutztes Flip Flop	61 %	24 %	47 %
Unbenutzte LUT	3 %	22 %	3 %
Benutzte LUT-FF Paare	35 %	53 %	48 %
Block-RAM	1	20	320 (76 %)
Maximale Taktrate			200 MHz
Vergleiche/s 64 bp			120 Giga
Erforderliche Datenrate			400 MBit/s

- Auf einem Virtex-5 lassen sich 100 Einheiten bei einer Taktrate von 100 MHz realisieren, was einer Leistung von 10 Giga Vergleiche/s entspricht.

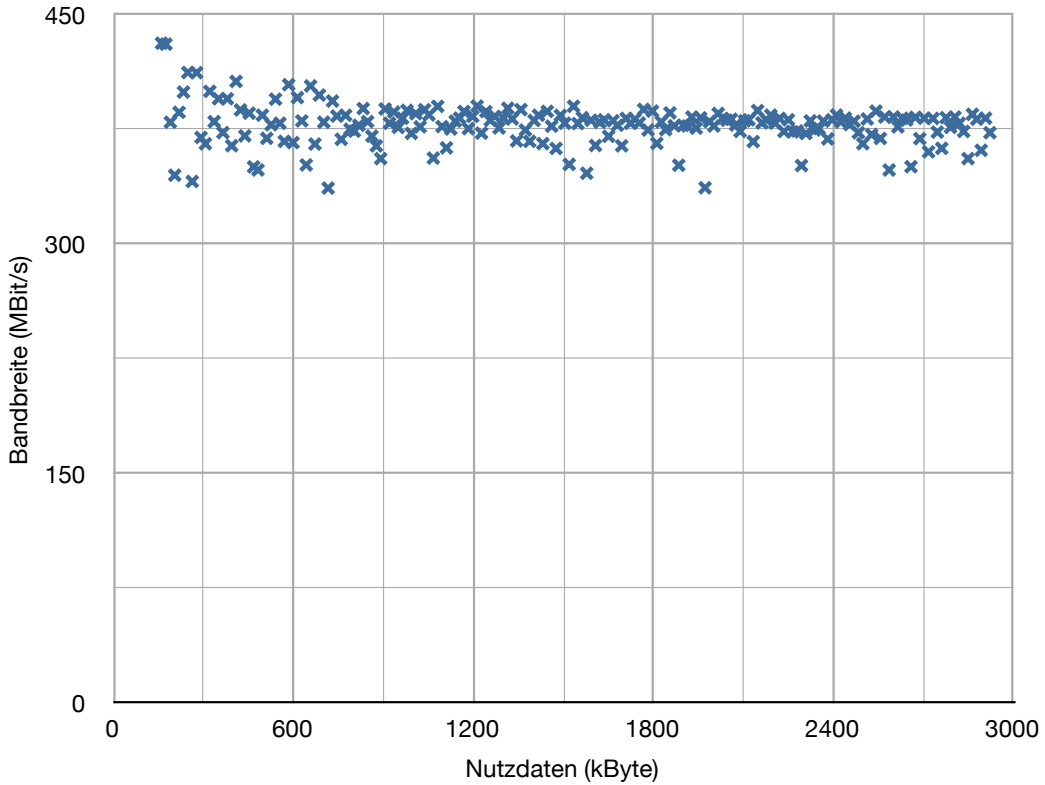
08 Anhang - FPGA

Host-Software III - 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

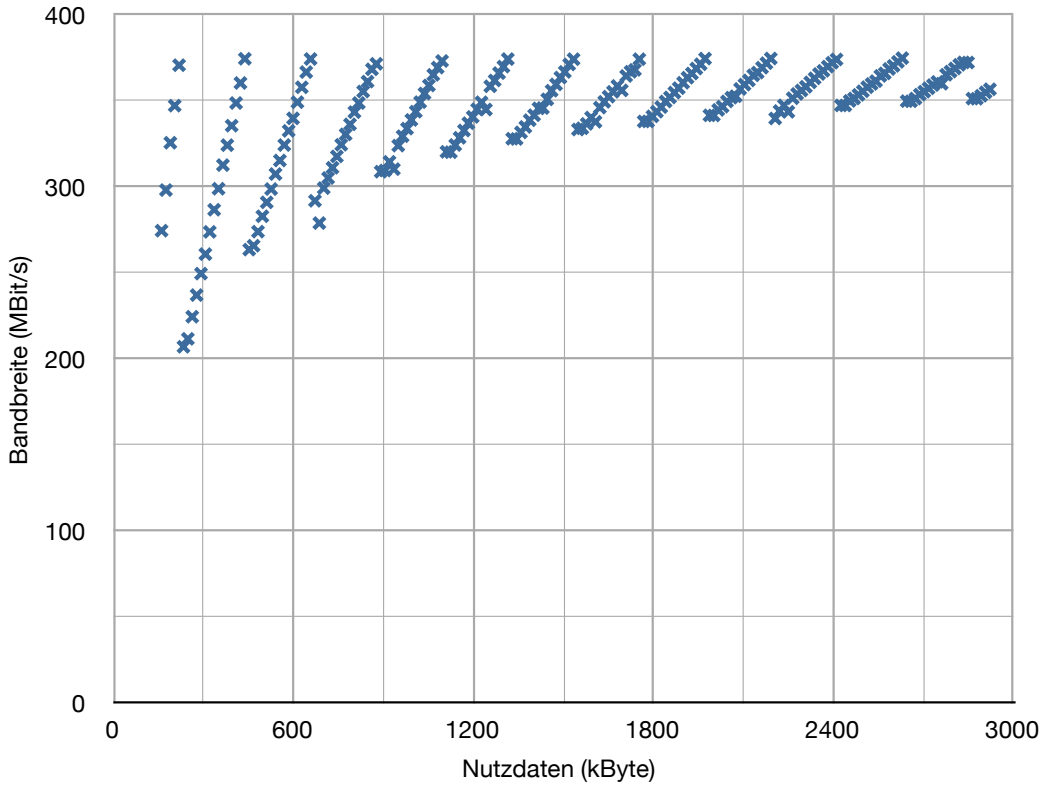
08 Anhang

Bandbreite - Virtex-6 (dynamische Flusskontrolle)



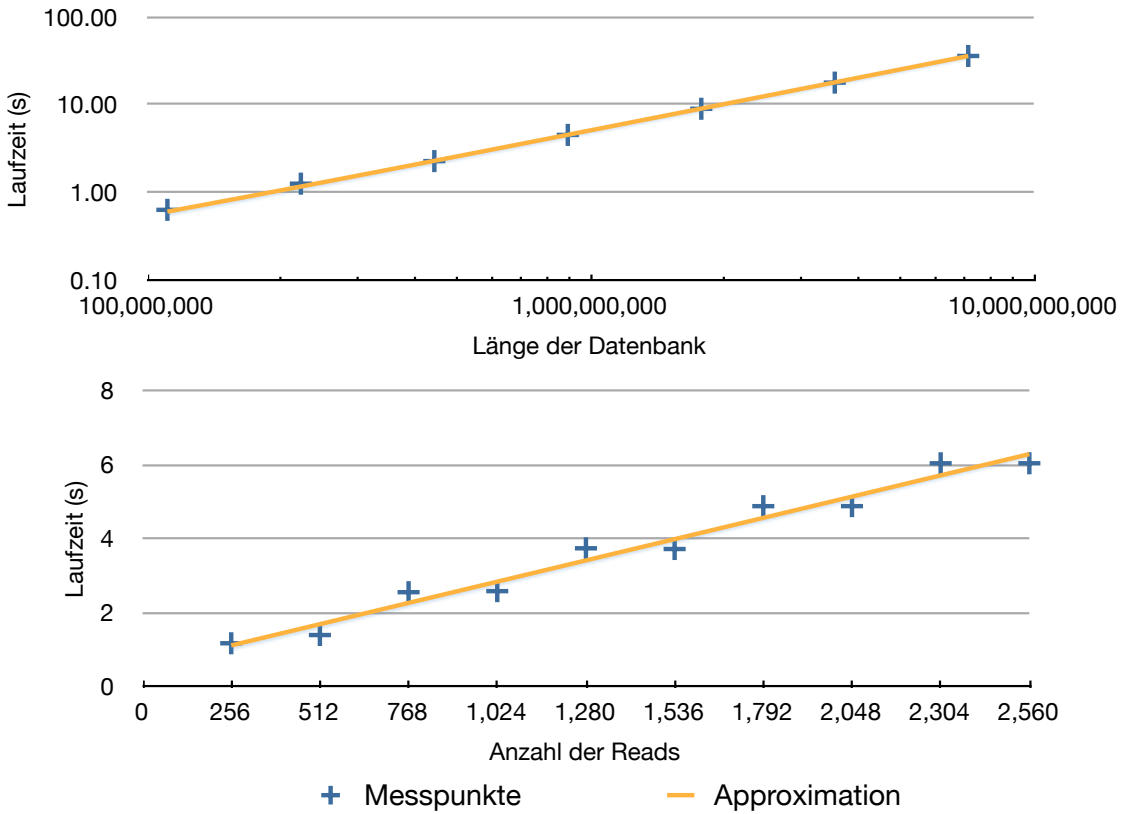
08 Anhang

Bandbreite - Virtex 6 (statische Flusskontrolle)



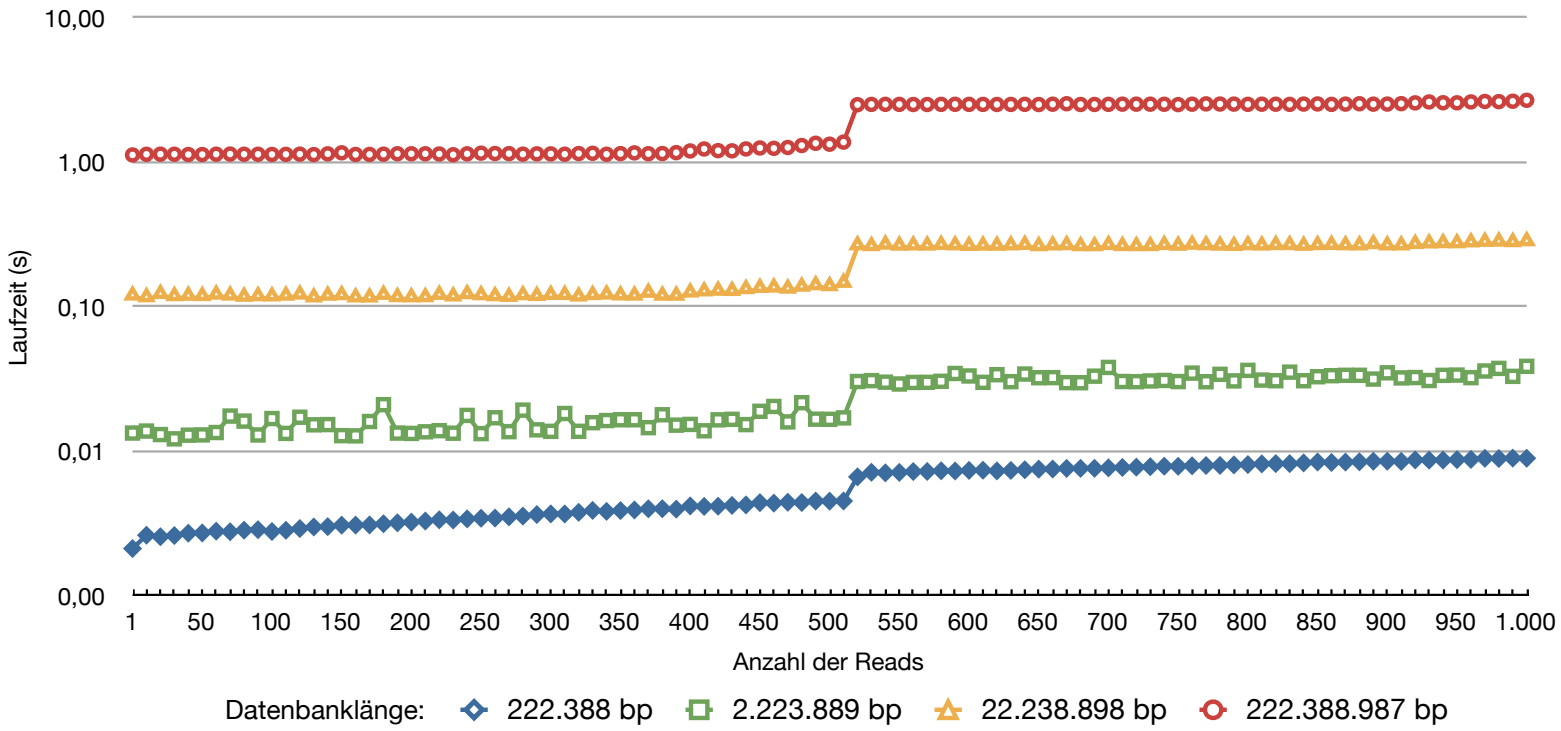
08 Anhang - FPGA

Skalierung I



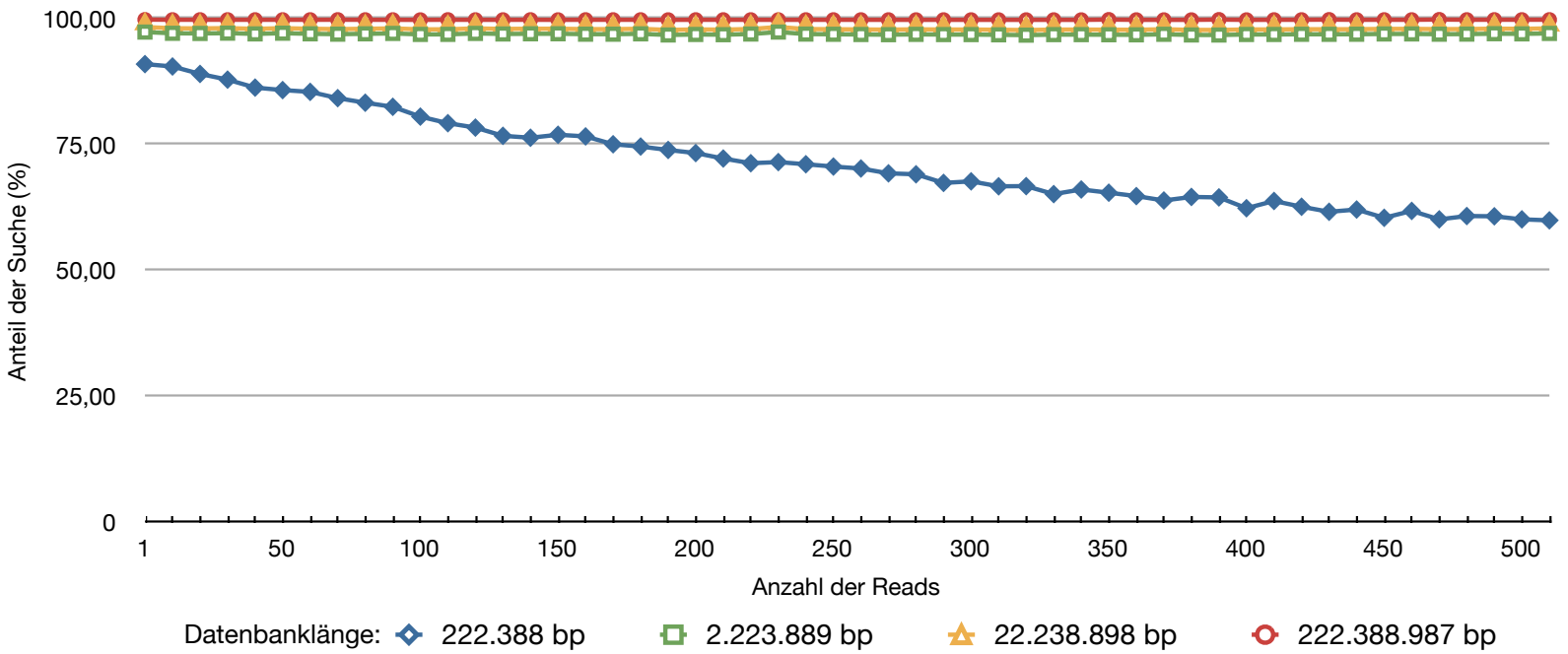
08 Anhang - FPGA

Skalierung II - Anzahl Reads



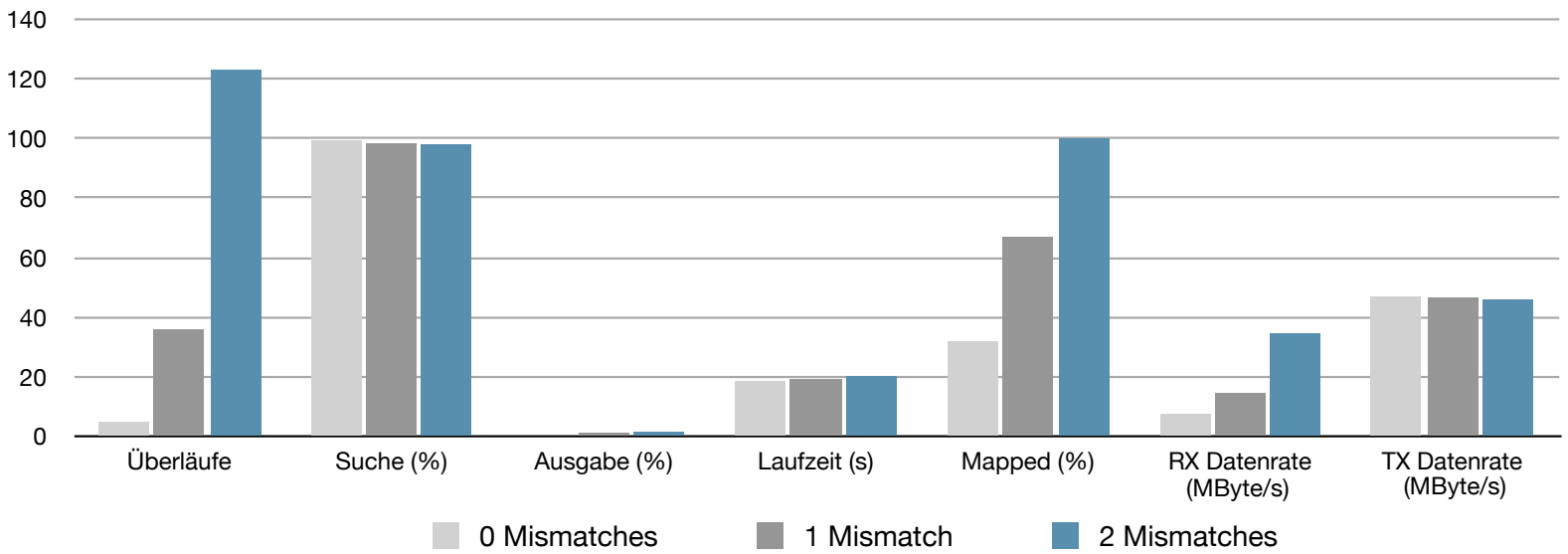
08 Anhang - FPGA

Skalierung III - Anteil der Suche



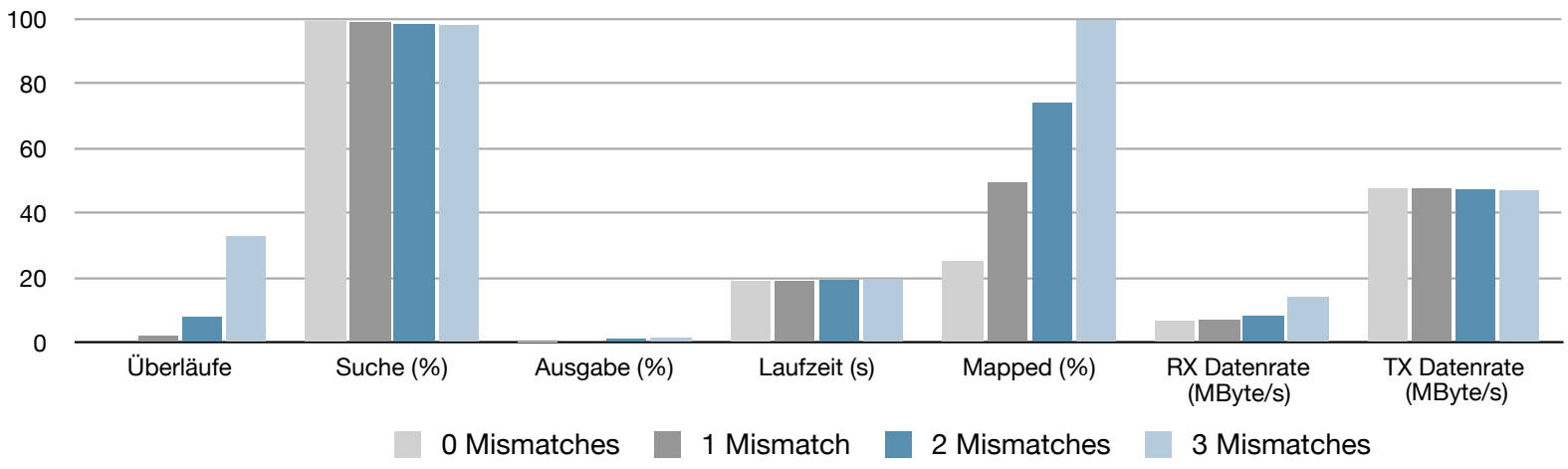
08 Anhang - FPGA

Verhalten bei Überläufen I - Kurze Reads



08 Anhang - FPGA

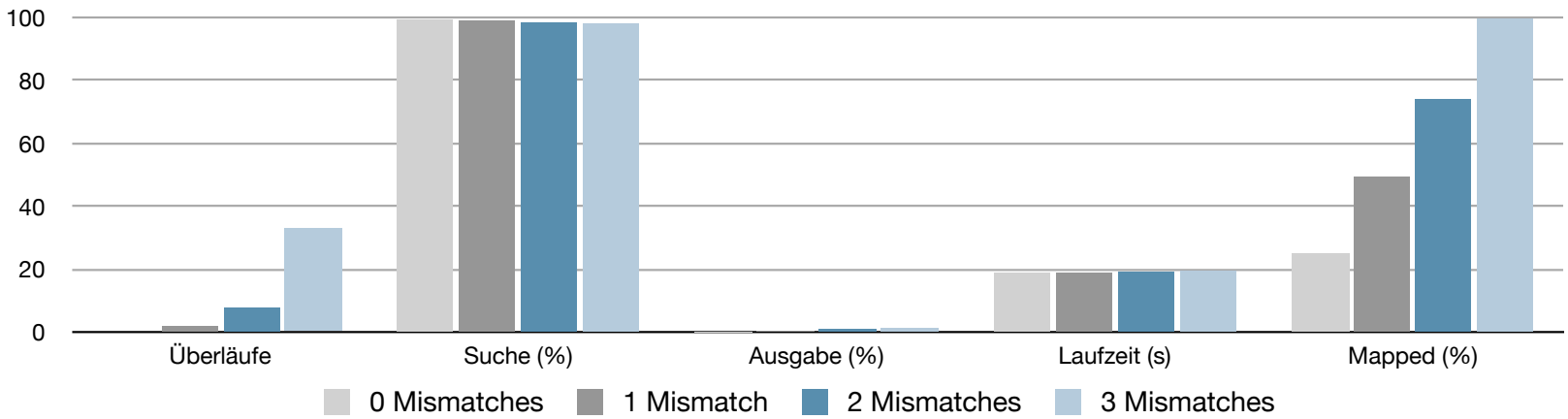
Verhalten bei Überläufen II - Lange Reads



08 Anhang - FPGA

Identifikation von Engpässen auf dem FPGA

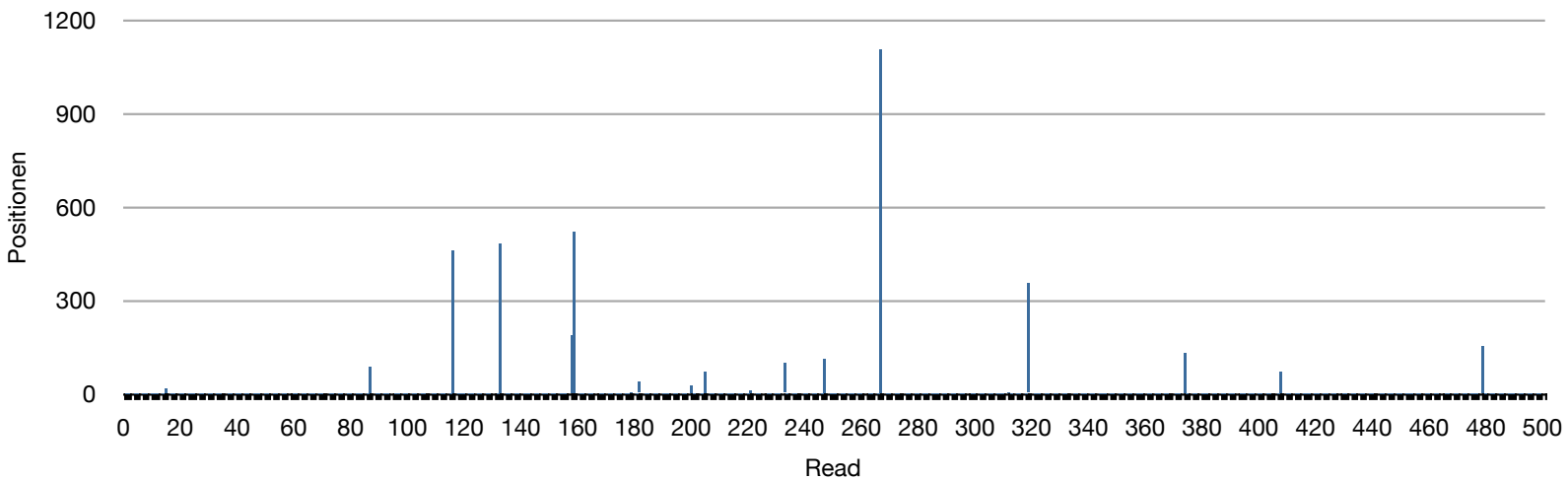
- Anteil der Laufzeit für Transformation, Übertragung und Einsammeln der Ergebnisse liegt bei 2 - 3 %.
- Erhöhung der Laufzeit durch Reaktion auf Überläufe ebenfalls gering.
- Starke Auswirkungen nur bei Datenbanken mit weniger als einer Million Basenpaaren.



Datenbank: Homo sapiens chromosome 1 (222.388.987 bp)
Reads: 100.000 mit 50 bp und 0-3 mismatches in diskreter Gleichverteilung

08 Anhang - FPGA

Verteilung der Reads

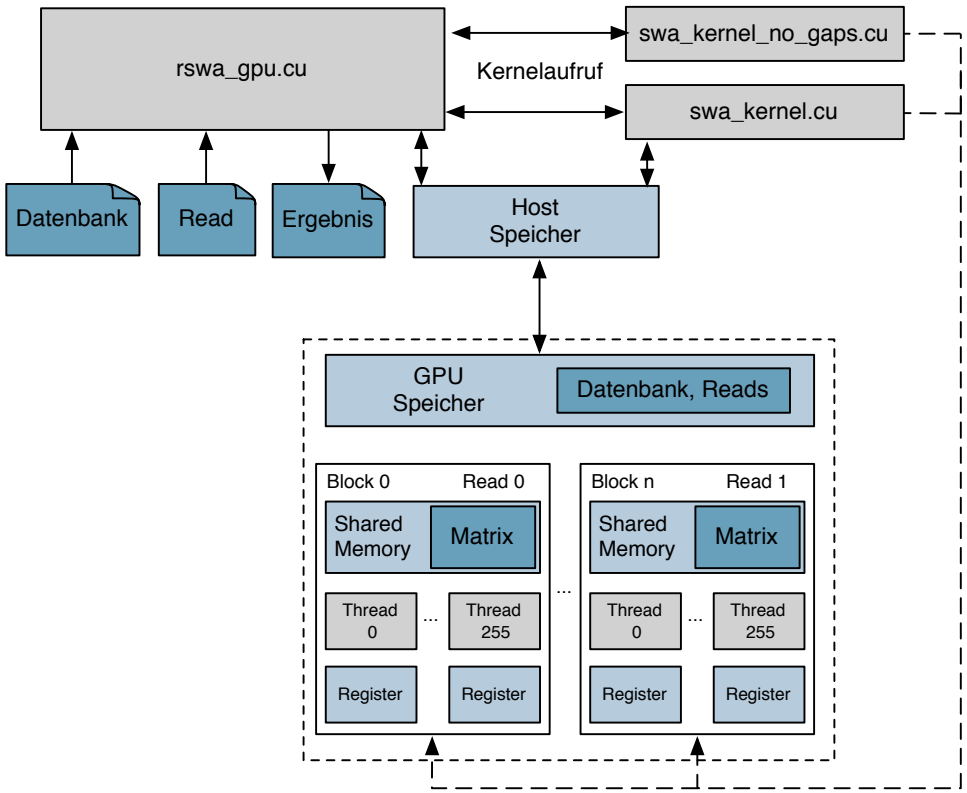


08 Implementierung - GPU

CUDA - Programm

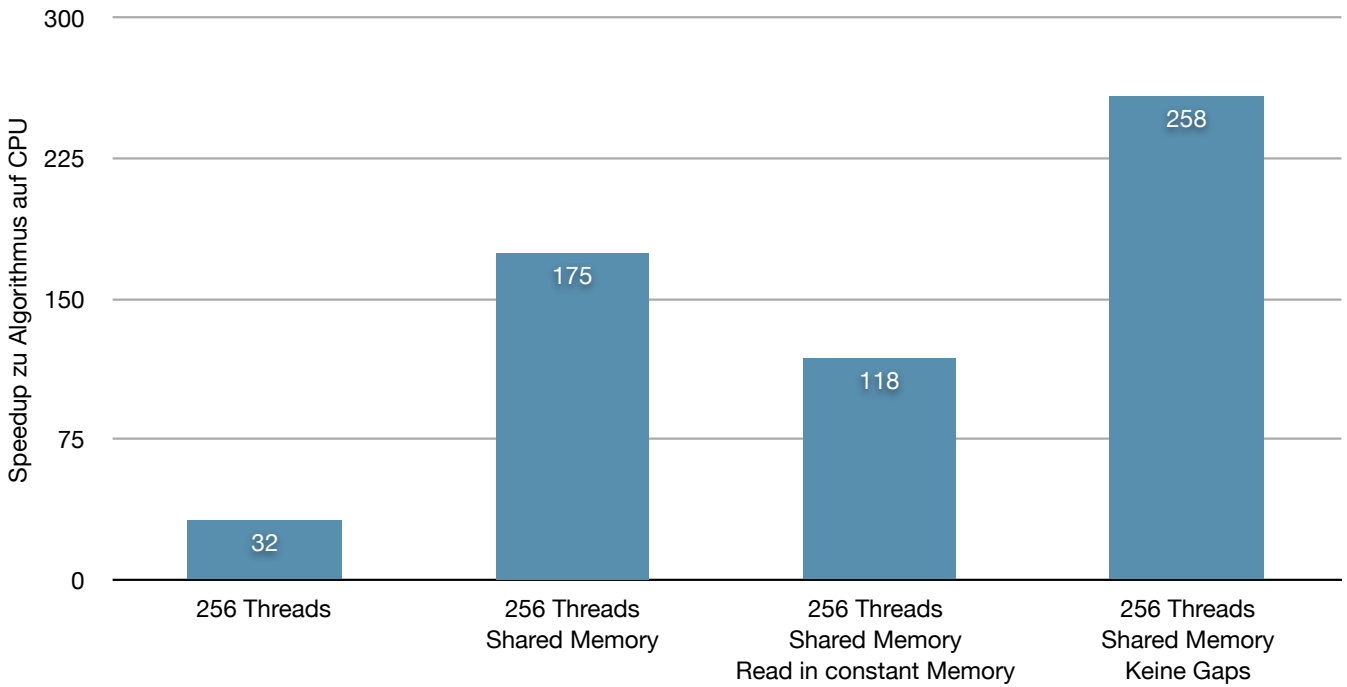
Verteilung der Daten:

- Datenbank im Grafikspeicher.
- Matrizen im gemeinsamen Speicher der Multiprozessoren.
- Reads zur Laufzeit in internen Registern der Threads.



08 Anhang - GPU

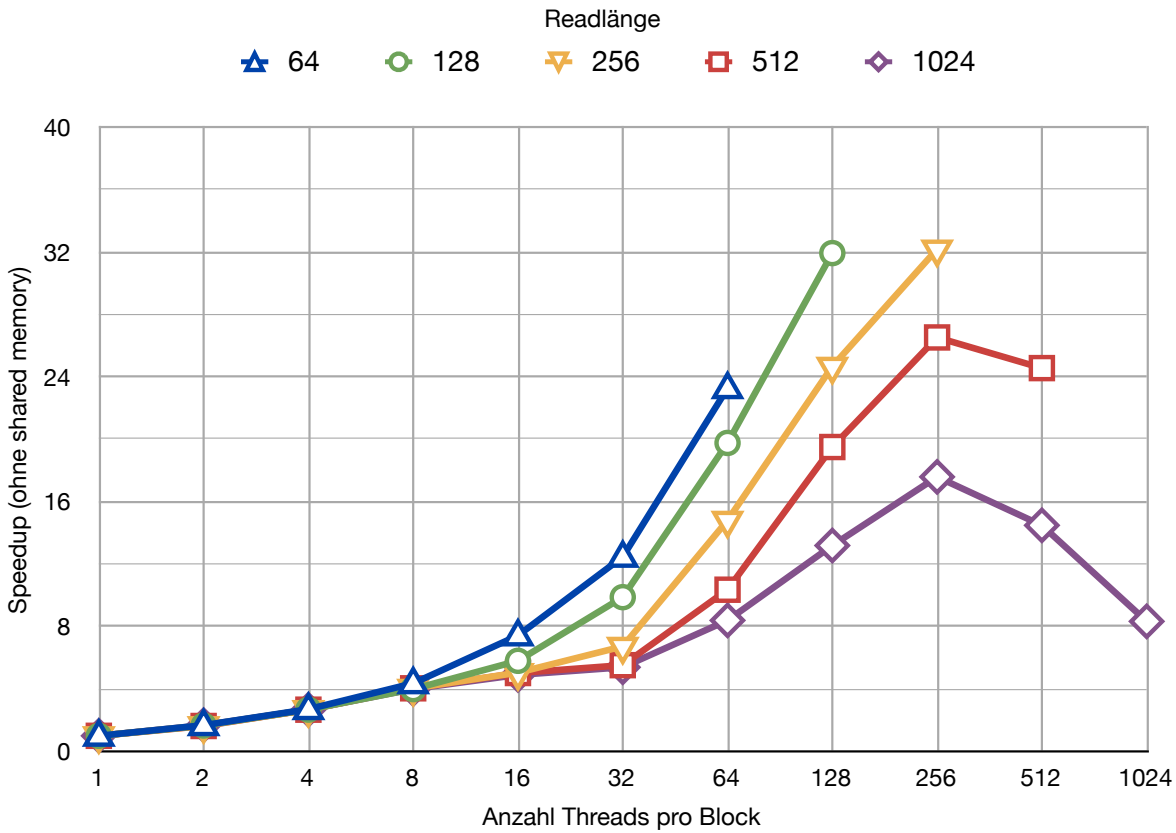
Optimierung



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

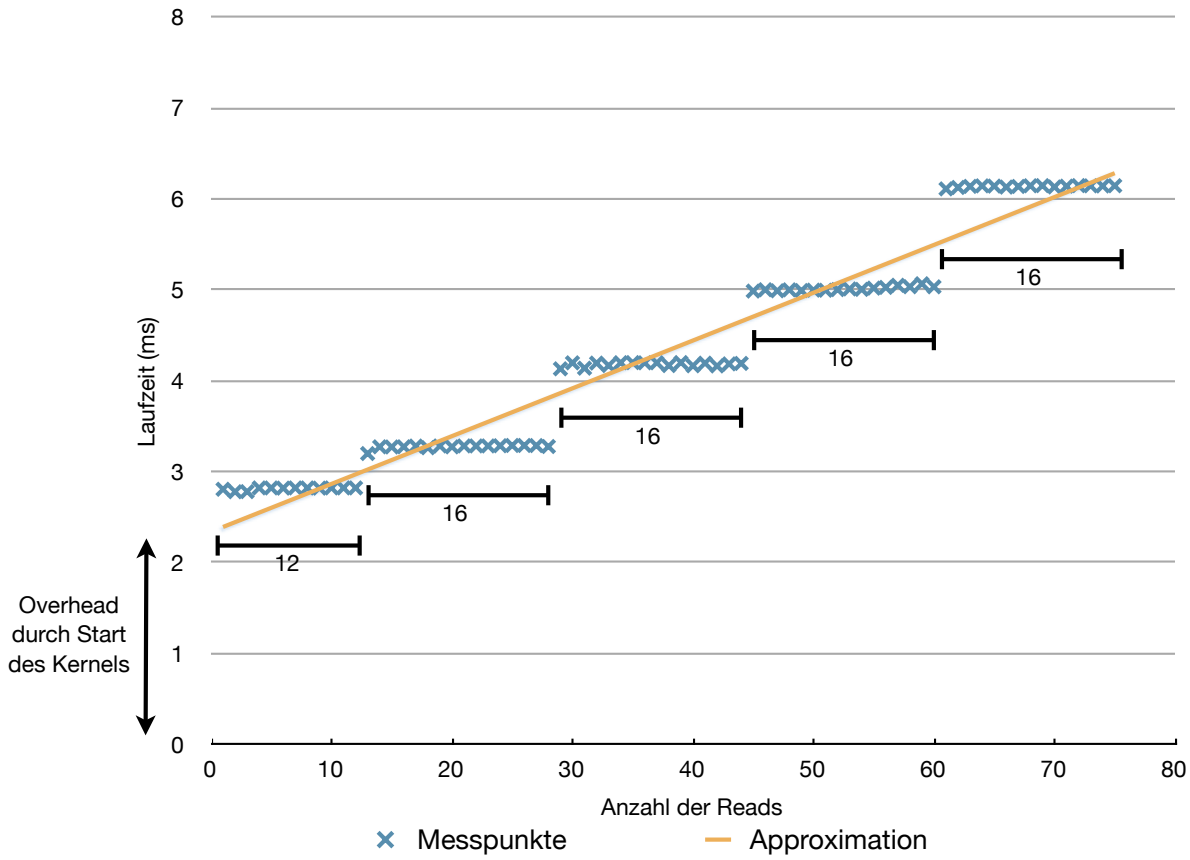
08 Anhang - GPU

Speedup für verschiedene Blockgrößen



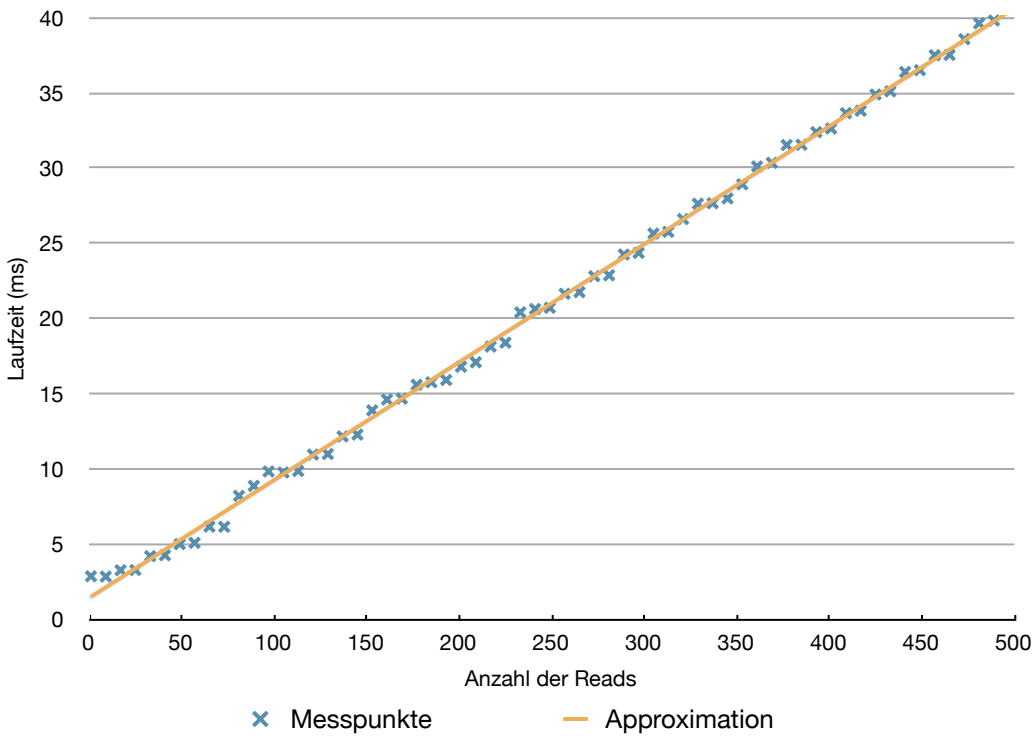
08 Anhang - GPU

Skalierung in Abhängigkeit der Anzahl der Reads I



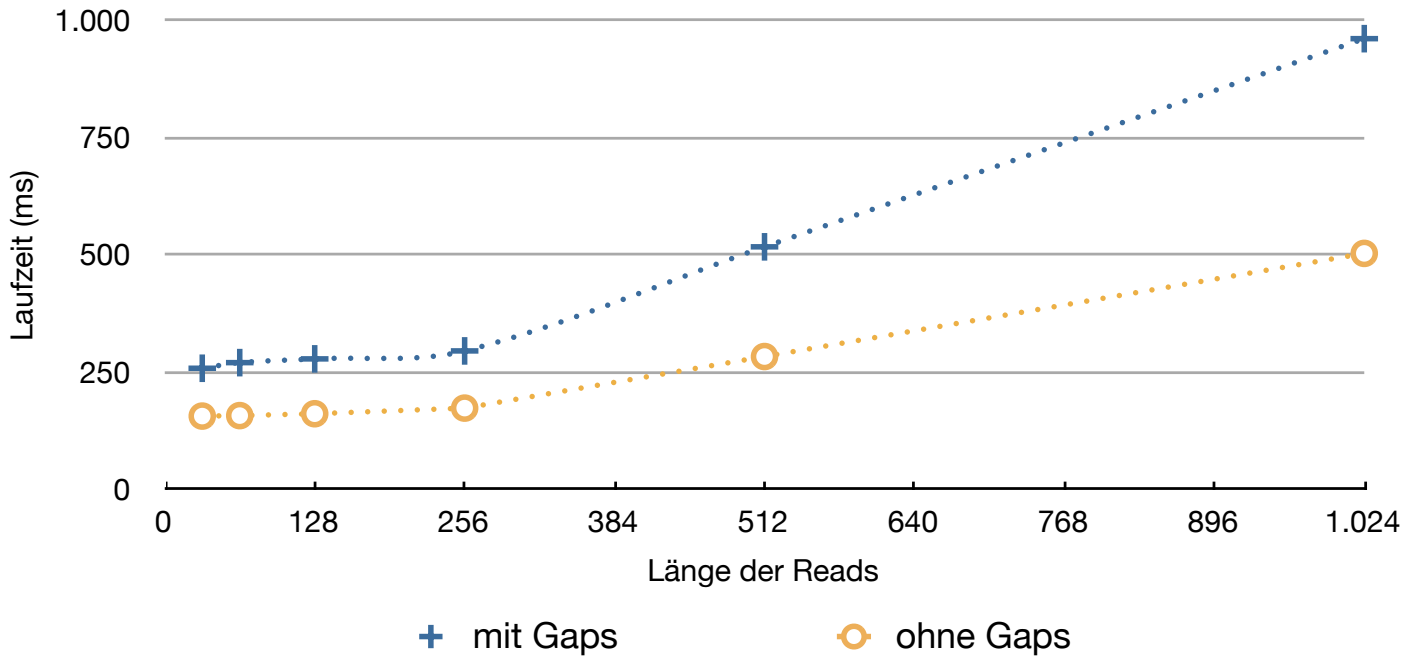
08 Anhang - GPU

Skalierung in Abhängigkeit der Anzahl der Reads II



08 Anhang - GPU

Skalierung bei unterschiedlicher Länge der Reads



08 Anhang - GPU

Leistung und Bandbreite

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