TECHNISCHE UNIVERSITÄT DRESDEN

DIGITAL SIGNAL PROCESSING USING CUDA

Digital Signal Processing using CUDA

Nico Wehmeier, Richard Pfeifer, Fabian Jung

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Dresden, 2.2.2014

Inhalt

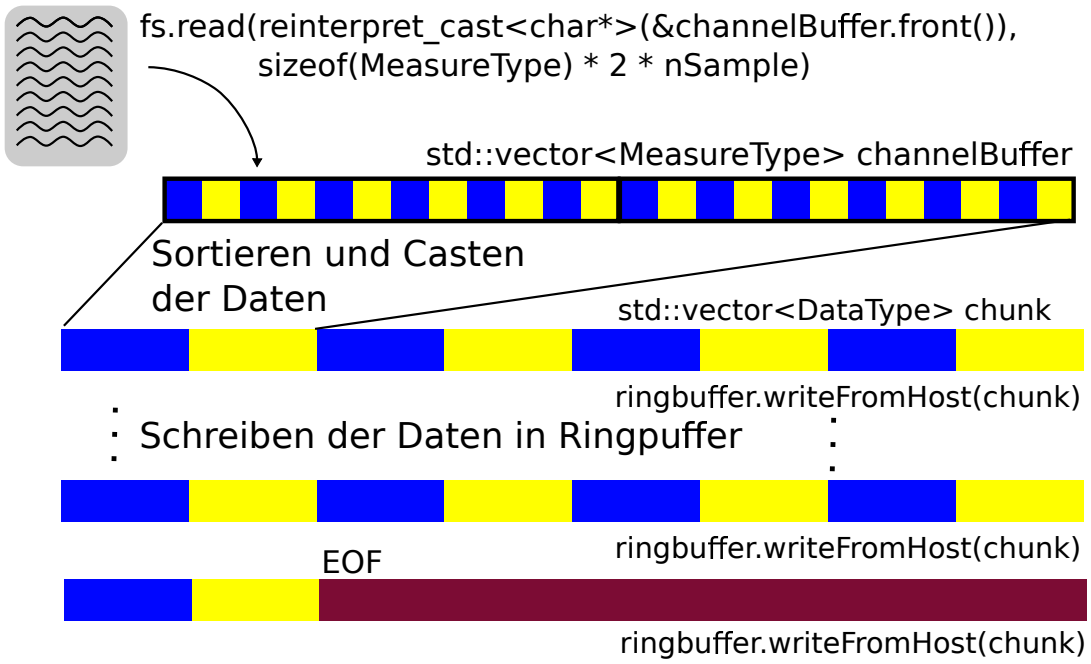
01 Aufgabenstellung

- Ausgangsitutation
  - Messgeräte erzeugen Datenstrom
  - Datenstrom nicht kontinuierlich
  - Serielle Implementierung
- Anforderungen
  - Portierung auf GPU
  - Hoher Datendurchsatz
  - Skalierbar auf bis zu 4 GPUs/Node

Überblick

- Host
  - Eingabe Datei auslesen
  - Daten zwischenspeichern
- Verwaltung Devices
  - Daten aus Puffer lesen
  - Zu den Devices streamen
  - Kernel starten
  - Ausgabe schreiben
- Levenberg Marquardt
  - Parameter einer Näherungsfunktion bestimmen
  - Markante Stellen (Anfangs-, Endwert, Maximum) ermitteln und zurückgeben

Host: Lesen der Daten



- Lesen von Daten zweier Kanäle
- Kanaltrennung
- Casten der Daten
- Sammeln mehrerer Signale zu Chunks
- Letzter Chunk wird mit Nullen gefüllt.

Host: Ringpuffer

- Messdatenfile → DataReader → Ringpuffer → GPU
- Speicher des Ringpuffer: Vektor dessen Typ per Template frei gewählt werden kann
- Host-GPU-Transfer wird wie folgt gelöst:


```
Type* ringbuffer.reserveTail()
    sem_trywai(&usage) !=0)
        return NULL
    sem_wait(&mtx)
    return &buffer.at(tail)

int ringbuffer.freeTail()
    tail = ++tail % bufferSize
    sem_post(&mtx)
    sem_post(&space)
    return 0
```

Daten vorhanden?

->externes Kopieren auf GPU möglich

Speicher kommunizieren

Verwaltung Devices

- Jedes Devices wird von einem Thread verwaltet
- Asynchrone Aufrufe
  - cudaMemcpyAsync
  - Kernel
  - Ermöglicht Pipeline
- Eigener Thread für Ausgabedatenstrom

Levenberg Marquardt (1)

- Eingabedaten
  - Samples
    - Compute Capability 1.x: ca. 800)
    - Compute Capability 2.0 oder höher: ca. 2500)
  - Interpolationsschritt
    - beliebige Dezimalzahl größer 0)
    - Interpolation durch Texture Memory)

Levenberg Marquardt (2)

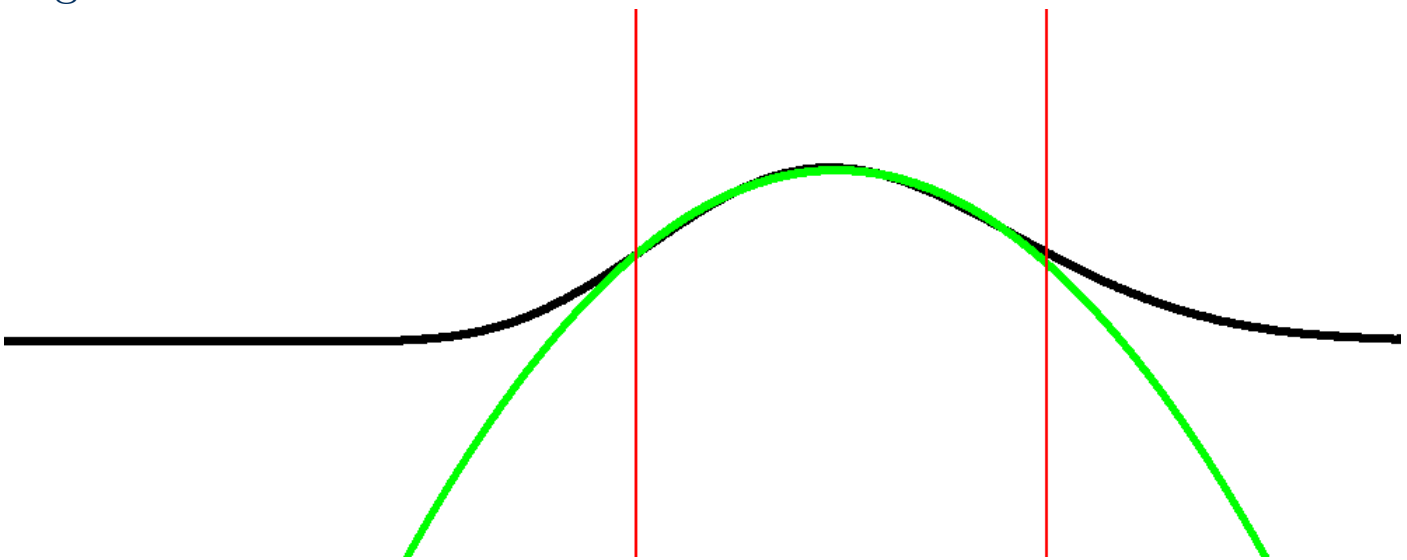
- Verarbeitung
  - eine Ausgleichungsrechnung pro Block
  - Grund: ca. das 5-fache der Sampleanzahl an Shared Memory benötigt (bei 1000 Samples ca. 20 kB)
  - Zugriff auf Samples durch Texture Memory
    - Shared Memory gespart
    - schnelle Interpolation möglich
  - Vorgehensweise
    - Anfangs- und Endwert ermitteln
    - abhängig vom Schwellwert Bereich festlegen
    - für den Bereich Näherungsfunktion ermitteln
    - Qualität durch Residuen und Maximum der Funktion ermitteln

Levenberg Marquardt (3)

- Ausgabedaten
  - 3 Parameter einer quadratischen Funktion: $a * x^2 + b * x + c$
  - Anfangs- und Endwert
  - Maximum
  - durchschnittliche Abweichung
  - Status (Fehler, Erfolg, Abbruch)

Levenberg Marquardt (4)

- Ergebnis: fitfunktion $= -0.550151 * x^2 + 654.15509 * x - 203167.921875$



Benchmark

GPUs	Datei	Zeit	Rate	Normiert
1	247MB	72.450s	3.41 MB/s	3.41 MB/s
2	247MB	38.141s	6.48 MB/s	3.24 MB/s
3	247MB	29.060s	8.50 MB/s	2.83 MB/s
4	247MB	20.828s	11.85 MB/s	2.96 MB/s
1	382MB	107.902s	3.54 MB/s	3.54 MB/s
2	382MB	57.537s	6.64 MB/s	3.32 MB/s
3	382MB	40.951s	9.33 MB/s	3.11 MB/s
4	382MB	29.885s	12.78 MB/s	3.20 MB/s

Skalierbarkeit

- Horizontal Skalierbar
- PCIe Bus noch nicht vollständig ausgelastet
- Reserven in der Implementierung des Ringbuffers

Danksagung

Vielen Dank an
Dr. Michael Bussmann, Axel Hübel und Rene Widera
für die Hilfe und Optimierungsvorschläge.