Fachpraktikum Parallele Programmierung, Wintersemester 2024/25

3. Aufgabe: Histogramm

*Johannes Becker, Norbert Baumstark, Stefan Butz*

**Aufgabenstellung**

Es sollten Routinen zur Erstellung eines Histogramms auf einer GPU implementiert und hinsichtlich ihrer Laufzeit untersucht werden. Dabei sollte gezählt werden, wie oft einzelne Bytewerte in einem Array von Bytes vorkommen. Das Hochzählen der Anzahlen in den Bins des Histogramms sollte durch atomare Operationen erfolgen. Es waren zwei Varianten zu untersuchen: zum einen das Hochzählen direkt im globalen Speicher; zum anderen sollten zunächst mehrere Instanzen der Bins im Shared Memory angelegt und diese erst abschließend zu einem einzigen Histogramm akkumuliert werden.

Die Eingabedaten waren nach Annahme Bytewerte zwischen 1 und 128. Es waren zwei Varianten des Histogramms zu erstellen, eine mit 128 Bins für die einzelnen Bytewerte sowie eine weitere mit jeweils einem Bin für die Buchstaben ‚A‘ bzw. ‚a‘ bis ‚Z‘ bzw. ‚z‘ und einem weiteren Bin für alle anderen Zeichen, d.h. insgesamt 27 Bins.

**Verzeichnisstruktur**

Auf der obersten Verzeichnisebene befinden sich zwei Dateien, nämlich ein Makefile sowie ein Python-Skript perform\_measurements.py, welches alle Messungen ausführt.

Es gibt folgende Unterverzeichnisse:

* src/ enthält den Quellcode.
* analysis/: enthält ein Jupyter-Notebook zur Auswertung der Messergebnisse.
* doc/ enthält diese Dokumentation.
* input\_data/ enthält die Testdaten, die der Aufgabenstellung beigefügt waren.
* nvidia/ enthält das mit den CUDA-Entwicklungstools mitgelieferte Beispielprogramm deviceQuery, welches Geräteinformationen über die GPU ausgibt.
* bin/ wird ggf. von make angelegt und dient als Zielverzeichnis für die Objektdateien sowie die ausführbaren Dateien deviceQuery und histogram.
* measurements/ wird ggf. von perform\_measurements.py angelegt und dient der Ablage der Geräteinformationen und Messergebnisse. In den abgegebenen Dateien sind die Messergebnisse enthalten, die dieser Dokumentation zugrunde liegen.

**Aufruf**

Nach Erstellen der ausführbaren Dateien durch

make

startet man durch

./perform\_measurements.py

die Messungen. Gegebenfalls sind die Parameter im Makefile auf die verwendete Architektur anzupassen.

Das Python-Skript ermittelt zunächst die Geräteinformationen durch Aufruf von bin/deviceQuery und ruft dann bin/histogram mit verschiedenen Kommandozeilenargumenten auf, um verschiedene Szenarien zu messen. Insbesondere wird bin/histogram angewiesen, pro Szenario und Kernel 100 Messungen vorzunehmen. Neben den der Aufgabenstellung beiliegenden Testdaten umfassen die gemessenen Datengrössen den Bereich von Bytes bis GiB in Zweierpotenzen. Die Ausführungsdauer des Python-Skripts ist also erheblich. Sollen aus Gründen der Zeitersparnis weniger als 100 Durchläufe je Szenario erfolgen, kann die Konstante N\_RUNS im Python-Skript entsprechend angepasst werden.

Die Geräteinformationen werden als Textdatei, die Messergebnisse im JSON-Format in einem Unterverzeichnis von measurements/ gespeichert. Das angelegte Unterverzeichnis trägt einen Zeitstempel als Name.