





Student/in:
Barras Simon

Zusammenfassung Bachelorarbeit 2023
INFORMATIK UND KOMMUNIKATIONSSYSTEME

DOZENT/IN: Frederic Bapst, Jean Hennebert	
AUFTRAGGEBER/IN: Lawrence Berkeley National Laboratory, Paolo Calafiura, Julien Esseiva	
INTERNER KONTAKT: iCoSys	
KURZZEICHEN PROJEKT: GPU Optimization Celeritas	INTERNE PROJEKT-NR.: B23ISC27
VERTIEFUNGSRICHTUNGEN: Informatik Software	EXPERTE/EXPERTIN: Dr. Baptiste Wicht
NACHHALTIGKEITSZIELE:  	

GPU-Optimierung in Celeritas, einer Simulationsbibliothek für Hochenergiephysik-Detektoren.

Da die Detektoren am LHC des CERN immer mehr Daten produzieren, können die Simulationen für die ATLAS- und CMS-Experimente nicht mehr mit reinen CPU-Tools durchgeführt werden. Der Einzug der GPUs in die Supercomputer eröffnet neue Entwicklungsperspektiven. Dies macht sich das Celeritas-Team zunutze, um eine Bibliothek von GPU-beschleunigten

Teilchenphysiksimulationen bereitzustellen.

Für diese Bachelorarbeit ging Simon Barras nach Berkeley in den USA, um am Lawrence Berkeley National Laboratory zu arbeiten. Dort arbeitete er mit dem Celeritas-Team zusammen, das in mehreren Labors in den USA, insbesondere in Oak Ridge, angesiedelt ist.

Celeritas

Bei diesem Projekt handelt es sich um eine Monte-Carlo-Teilchentransportbibliothek für die Simulation von Detektoren der Hochenergiephysik. Es ist motiviert durch die massiven Rechenanforderungen der ATLAS- und CMS-Experimente im LHC am CERN. Das Ziel ist es, Geant4 zu ersetzen, das bisherige Werkzeug für die Simulation von Teilchen. Celeritas wurde entwickelt, um auf Supercomputern.

GPU

Grafische Prozessoreinheiten (GPU) sind Chips, die als Single Instruction, Multiple Data (SIMD) oder Signe Instruction, Multiple Threads (SIMT) definiert sind. Diese Karten sind effizient, wenn eine kleine

Anzahl von Operationen mit einer großen Datenmenge durchgeführt werden muss.

Um ein Programm mit einer GPU zu entwickeln, wird die Plattform der GPU benötigt, um festzulegen, welcher Teil des Codes auf dem Gerät ausgeführt werden muss. Die Plattform für Nvidia heißt zum Beispiel CUDA. Wenn wir einen Kernel starten, legen wir die Anzahl der Blöcke und die Anzahl der Threads pro Block fest.

Dormand Prince

Runge Kutta Dormand Prince (RKDP) ist ein Solver. gewöhnlicher Differentialgleichungen der 4. und 5. Ordnung und der fünften Ordnung. Er ist definiert durch den Koeffizienten der Tabelle von Butcher definiert. Die Implementierung erfolgt in neun Schritten durchgeführt. Die ersten sechs Schritte bestehen aus den Zwischenschritt mithilfe der Gleichung zu berechnen und den aktuellen Zustand zu aktualisieren. Die letzten drei Schritte bestehen hauptsächlich aus der Berechnung des Fehlers und den Zustand des resultierenden Punktes.

Design

Es ist wichtig, den Code gut zu verstehen, um einen Weg zur Parallelisierung zu finden, auch wenn RKDP iterativ ist und es unmöglich erscheint. Die Implementierung der Methode im Projekt besteht aus drei Arten von Code. Die erste Kategorie ist die Berechnung des Zwischenzustands mit der Gleichung. Der zweite Teil ist der Code, der keine Abhängigkeiten hat. Die letzte Kategorie ist die Aktualisierung des Zustands in Form von zwei 3-Dim-Vektoren. Dieser Teil kann



Student/in:
Barras Simon

Zusammenfassung Bachelorarbeit 2023
INFORMATIK UND KOMMUNIKATIONSSYSTEME

in Teilaufgaben aufgeteilt werden, die parallelisiert werden können. Abbildung 1 zeigt ein Profil der Ausführungszeit für jeden Schritt und jeden Typ. Anhand dieser Ergebnisse und unter der Annahme, dass die neue Implementierung auf vier Threads verteilt wird, ist es möglich, die Aufgaben so zu verteilen, dass die Ausführungszeit der Methode reduziert wird. Abbildung 2 zeigt ein Beispiel für eine optimale Arbeitslast.

Implementierung

Es wurden zwei Implementierungen vorgenommen. Beide haben vier Threads, wobei der erste als Controller konzipiert ist und den sequentiellen Teil berechnet, wenn die drei Worker die Vektormultiplikation berechnen. Der Unterschied zwischen den beiden Implementierungen liegt in der Nutzung des gemeinsamen Speichers. Dieser Speicher ist schneller als der globale Speicher, aber seine Größe ist auf wenige Kilobytes pro Block begrenzt. Abbildung 3 zeigt zwei Diagramme, die die Kommunikation innerhalb der Threads veranschaulichen.

Ergebnisse

Die Ergebnisse der beiden Implementierungen sind ziemlich gut, wenn nur ein Block verwendet wird. Sie beschleunigen den Prozess um durchschnittlich 150%, können aber weniger Partikel verfolgen als die alte Version. Die Version mit globalem Speicher ist durch die Anzahl der Threads pro Spur begrenzt und die Version mit gemeinsamem Speicher durch die Größe des Speichers selbst. Abbildung 4 zeigt den Geschwindigkeitszuwachs und die Beschränkungen.

Da Celeritas eine große Anzahl von Partikeln verfolgen will, läuft es auf mehr als einem Block. Abbildung 5 zeigt die Entwicklung der Zeit in Abhängigkeit von der Anzahl der Spuren. Die Ergebnisse sind ziemlich schlecht, da die neue Version einige Schwierigkeiten hat, zu skalieren. Jedes Mal, wenn 84 Blöcke angefordert werden (ca. 15'000 Spuren für die neuen Implementierungen), übersteigt die Menge an Ressourcen, die von der GPU angefordert wird, die Menge, die auf einmal läuft. Diese Überschreitung wird durch den Sprung in der Grafik veranschaulicht.

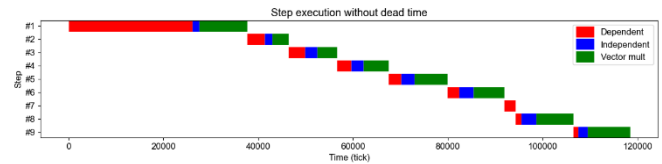


Figure 1 Ergebnisse der RKDP-Profilierung

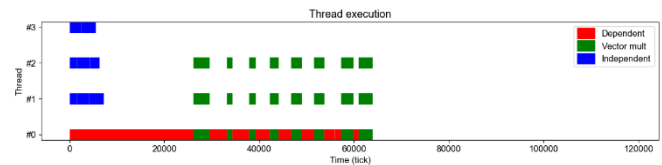


Figure 2 Optimal task scheduling

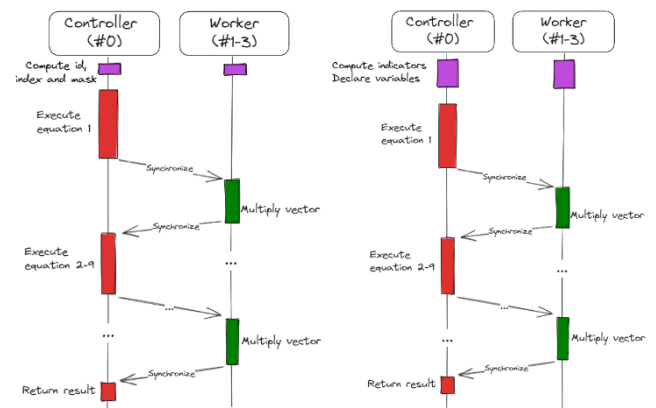


Figure 3 Implementierung mit jeweils globalem und gemeinsamem Speicher

Evolution of the different implementations with the number of tracks using 1 block

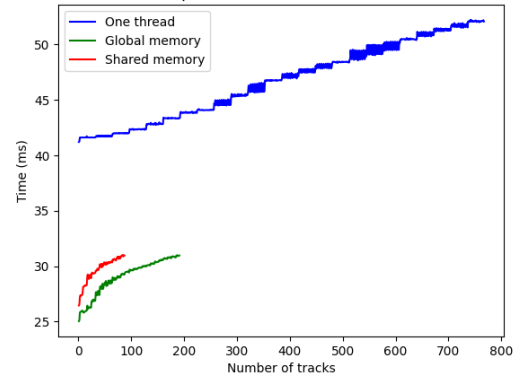


Figure 4 Performance with one block

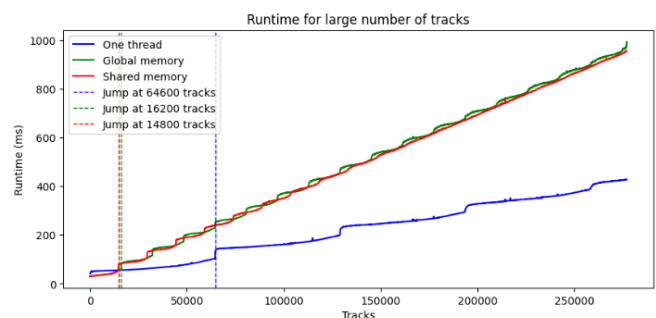


Figure 5 Leistung mit Multiblock