Kernkräfte als Videospiel

Von Hartmut Wittig

Die kommerzielle Bedeutung von Computeranimation und Videospielen ist in den vergangenen Jahren enorm gewachsen. Im Zuge dessen stieg auch der Bedarf an immer realistischeren Visualisierungen, was zur Entwicklung spezieller Grafikprozessoren geführt hat, deren Rechengeschwindigkeit um ein Vielfaches größer ist als die von herkömmlichen Prozessoren. Mainzer Physiker versuchen nun, die enorme Rechenleistung solcher Grafikprozessoren für wissenschaftliche Anwendungen zu nutzen.

Computergrafik ist aus vielen Bereichen des öffentlichen und wissenschaftlichen Lebens nicht mehr wegzudenken: Moderne CAD-Verfahren („Computer Aided Design") haben das traditionelle Reißbrett abgelöst. Die computergestützte Visualisierung physikalischer Vorgänge dient nicht allein nur dem detaillierten Sichtbarmachen eines Prozesses, sondern ergänzt die abstrakte Beschreibung durch die Mathematik. Auch in der Unterhaltungsindustrie hat sich die Computergrafik fest etabliert: Computeranimationsfilme, virtuelle Welten („Second Life") und Spielekonsolen (Nintendo, Playstation, Xbox etc.) findet man heute in fast jedem Kinderzimmer. Realistische Visualisierungen erfordern allerdings einen riesigen Rechenaufwand, der mit den üblichen Mikroprozessoren, wie sie in Arbeitsplatzrechnern und Laptops eingebaut sind, nicht erreicht werden kann. Daher wurden entsprechend optimierte Grafikprozessoren (GPUs) entwickelt. Der rasante technologische Fortschritt in der computergestützten Bildgenerierung der vergangenen Jahre ist dabei nicht zuletzt durch die enormen Umsätze befeuert worden, die mit Animationsfilmen und Spielekonsolen erzielt werden konnten.

Wo so viel Rechenleistung zur Verfügung steht, stellt sich die Frage, ob sich solche Grafikprozessoren auch für wissenschaftliche Rechnungen einsetzen lassen. Hierbei steht also nicht die Nutzung der GPUs für die Bildgenerierung, sondern das Anzapfen ihrer schieren Prozessorleistung für besonders rechenzeitintensive wissenschaftliche Anwendungen im Vordergrund. Eine solche Anwendung ist die numerische Untersuchung und Simulation der Kernkräfte. Die sichtbare Materie des Universums besteht neben den Elektronen der Atomhülle im Wesentlichen aus den Protonen und Neutronen des Atomkerns. Die kleinsten bekannten Bausteine der Materie sind jedoch die Quarks, die durch den Austausch von Gluonen zusammengehalten werden (engl. „glue" = Leim). Zur Beschreibung der Kräfte zwischen Quarks und Gluonen kennt man seit den 1970er Jahren eine Theorie, die sogenannte Quantenchromodynamik (QCD). Die Frage, ob sich die experimentell bestimmten Eigenschaften der Protonen und Neutronen aus der QCD herleiten lassen, ist allerdings nach wie vor eine wissenschaftliche Herausforderung, denn die komplexe mathematische Struktur der QCD hat sich bisher allen Lösungsverfahren mit Bleistift und Papier widersetzt.

Der spätere Nobelpreisträger Ken Wilson hat bereits 1974 die Grundlagen für eine numerische Behandlung der QCD auf Großrechnern geschaffen.[1] Er schlug damals vor, die QCD auf einer diskretisierten Raumzeit zu formulieren, ähnlich einem Kristallgitter. Die einzelnen Gitterplätze werden dabei mit den Quarkfeldern besetzt, während die Gluonen auf den Verbindungslinien (Kanten) zwischen einzelnen Gitterplätzen sitzen. Die Berechnung physikalischer Größen, wie beispielsweise der Masse des Protons, kann in dieser Formulierung durch eine sogenannte Monte- Carlo-Simulation erfolgen, ein Verfahren, das häufig auch in der Festkörperphysik angewendet wird.

Es fällt nicht schwer zu verstehen, dass eine präzise Berechnung physikalischer Größen nur dann gelingen kann, wenn das Gitter möglichst feinmaschig ist, denn schließlich ist die reale Raumzeit keine diskrete Punktmenge. Will man ein Proton in ein genügend großes Volumen einsperren und gleichzeitig den Gitterabstand möglichst klein halten, braucht man eine große Zahl von Gitterplätzen, was natürlich mit einem hohen Rechenaufwand einhergeht. Mit der derzeit verfügbaren Rechner-Hardware lassen sich Systeme mit zehn Millionen Gitterpunkten behandeln, wobei der Gitterabstand etwa 1/20 der Ausdehnung eines Protons beträgt.

Derartige Systeme sind zu groß, als dass sie von einzelnen Prozessoren bewältigt werden könnten. Daher muss man Parallelrechner einsetzen: Hierbei wird das Gesamtsystem in viele verschiedene Untergitter aufgeteilt, die jeweils in den Hauptspeicher eines einzelnen Prozessors passen. Allerdings sind diese Untergitter nicht unabhängig voneinander, so dass gelegentlich die Information eines Untersystems mit der des Nachbarprozessors über ein schnelles Kommunikationsnetzwerk ausgetauscht werden muss. Wie effizient ein solches Verfahren ist, hängt ganz wesentlich davon ab, wie gut Netzwerkgeschwindigkeit und Prozessorleistung aufeinander abgestimmt sind. Eine Maschine, bei der die Balance zwischen verschiedenen Hardware-Komponenten realisiert ist,

Klaus Weindel

Abb. 1: Das Cluster „Wilson" am Institut für Kernphysik.

ist das kürzlich eingeweihte Cluster „Wilson" am Institut für Kernphysik (Abb. 1). Dieses Cluster besteht aus insgesamt 2.240 Prozessorkernen, die über ein Infiniband-Netzwerk miteinander gekoppelt sind.[2] Die Rechengeschwindigkeit eines einzelnen Prozessors für Anwendungen, wie sie in der Gitter-QCD typisch sind, beträgt dabei 1,65 GFlop/s (engl.: Giga-floating-point-operations per second = Milliarden Rechenoperationen pro Sekunde). Damit erreicht unser Anwendungscode zirka 20 Prozent der theoretisch möglichen Rechenleistung eines Prozessorkerns. Das gesamte Cluster hat eine Leistung von 3.700 GFlop/s. Berücksichtigt man die Anschaffungskosten von 1,1 Millionen Euro, so muss man rund 300 Euro pro GFlop/s bezahlen. Damit liegt „Wilson" in der Spitzengruppe, was Kosteneffizienz betrifft.

Das „Wilson"-Cluster und auch die anderen Hoch- und Höchstleistungsrechner an Universitäten und nationalen Rechenzentren basieren auf Prozessoren (CPUs), die sich nur wenig von denen unterscheiden, die man in handelsüblichen Arbeitsplatzrechnern findet. Da der Rechenzeitbedarf je nach wissenschaftlicher Fragestellung und erforderlicher Genauigkeit die derzeitigen Kapazitäten um mehrere Größenordnungen übersteigen kann, liegt es nahe, die hohen Rechenleistungen von Grafikprozessoren für die Gitter-QCD nutzbar zu machen. In einem gemeinsamen Projekt mit der Arbeitsgruppe von Prof. Elmar Schömer vom Institut für Informatik untersucht der Arbeitskreis „Gitter-QCD" am Institut für Kernphysik derzeit, inwieweit sich Grafikprozessoren für typische QCD-Anwendungen eignen.

Als Standard-„Benchmark" dient hierfür eine Matrix-Vektor-Multiplikation. Im Rahmen der Gitter-QCD beschreibt die Matrix die physikalische Wechselwirkung zwischen den Quark- und Gluonfeldern. Die Dimension der Matrix ist riesig: Die Zahl der Spalten ist gleich der Zahl aller Gitterpunkte multipliziert mit weiteren Quantenzahlen, die den Zustand eines Quarks charakterisieren. Benutzt man das oben genannte Beispiel von zehn Millionen Gitterpunkten, so hat die Matrix 1,44 mal 10^{16} Elemente. Allerdings sind die meisten davon Null, denn die Kräfte zwischen den Quarks wirken entweder am gleichen Ort oder nur zwischen benachbarten Gitterpunkten. In der Fachsprache der Mathematik nennt man ein solches Objekt eine dünn besetzte Matrix. Es gibt verschiedene Möglichkeiten, eine konkrete Wahl dieser Matrix der Quark-Gluon-Wechselwirkung anzugeben. In unserem Projekt benutzen wir die ursprüngliche Formulierung von Ken Wilson, die unter dem Namen „Wilson-Dirac-Operator" bekannt ist. Im Laufe einer typischen numerischen Simulation der Gitter-QCD benötigen die Anwendungen des „Wilson-Dirac-Operators" mehr als 70 Prozent der gesamten Rechenzeit. Folglich konzentrieren sich alle Versuche, numerische Simulationen der Gitter-QCD zu beschleunigen, auf die Optimierung des Wilson-Dirac-Operators.

Ein wesentliches Architekturmerkmal von Grafikprozessoren ist die große Anzahl von Prozessorkernen, die auf den derzeit gängigen Plattformen zwischen 128 und 240 variieren kann. Zwar gibt es auch bei herkömmlichen CPUs einen deutlichen Trend zu Mehrkernprozessoren, doch nehmen sich die han-

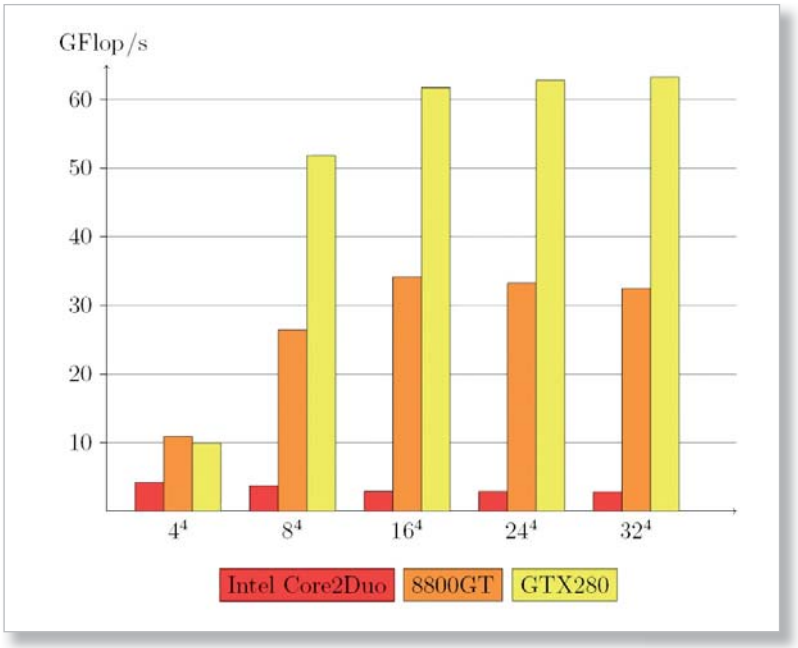

Abb. 2: Erzielte Rechengeschwindigkeit bei der Multiplikation mit dem „Wilson-Dirac-Operator" für verschiedene Gittergrößen. Verglichen werden herkömmliche Prozessoren (rot) und verschiedene Grafikkarten (orange und gelb). GFlop/s = Milliarden Rechenoperationen pro Sekunde.

delsüblichen Intel- oder AMD-Chips mit typischerweise vier Kernen recht bescheiden neben den GPUs aus. Obwohl sich GPUs auch durch eine sehr hohe Speicherbandbreite auszeichnen, die um mehr als eine Größenordnung höher als bei normalen Prozessoren sein kann, erweist sich der Zugriff auf den Hauptspeicher als eines der größten Hindernisse für eine effiziente Programmierung.

Wer sich mit der Programmierung von Grafikprozessoren befasst, stellt sehr schnell fest, dass der damit verbundene Aufwand erheblich größer ist als für herkömmliche Prozessoren. Der Grund liegt darin, dass Standard-Programmiersprachen wie C oder C++ (von Fortran ganz zu schweigen) nicht zur Verfügung stehen. Bis vor Kurzem erforderten daher selbst einfachste Operationen, wie die Addition oder Multiplikation zweier Zahlen, viele Zeilen Programmcode. Glücklicherweise haben einige Hersteller damit begonnen, Programmierumgebungen für ihre Grafik-Hardware anzubieten, die denen einer Standardsprache gleichen. Beispielsweise hat die Firma Nvidia die Programmierplattform CUDA („Compute Unified Device Architecture") entwickelt, die auch die Basis für unser Projekt bildet. Die Syntax von CUDA ist ähnlich der Programmiersprache C, erlaubt jedoch, einzelne Hardware-Komponenten des Grafik-Prozessors direkt anzusteuern. Trotz dieser Vereinfachung braucht es einige Übung, bis es gelingt, einen effizienten Code für GPUs zu schreiben. Grafikprozessoren verfügen über besondere Mechanismen zum effizienten Zugriff auf zusammenhängende Speicherbereiche. In der Computergrafik ergibt sich die Notwendigkeit hierfür aus der Verwendung von sogenannten Texturen (Pixelbildern), durch die der Detailgrad von dreidimensionalen Bildern erhöht wird. Die Verwendung von Texturen stellt sicher, dass der relativ langsame Zugriff auf den Hauptspeicher die Rechenleistung nicht drastisch vermindern kann. Ein wesentlicher Teil der Programmierarbeit im Rahmen unseres Projekts bestand daher in der Übertragung der Datenzugriffe für die Quark- und Gluon-Felder der QCD auf den Textur-Speicher.

Ein konkretes Beispiel, das den Unterschied zwischen der Optimierung von CUDA-Code im Vergleich zu Simulationsprogrammen in C illustriert, ist der Speicherzugriff auf das Gluon-Feld. Gluon-Felder werden durch unitäre 3x3 Matrizen dargestellt. Eine solche Matrix besteht aus neun komplexen Zahlen, was insgesamt 18 reellen Zahlen entspricht. Da die Matrix unitär ist, sind die Vektoren, die den drei Zeilen entsprechen, nicht linear unabhängig. Auf einer herkömmlichen CPU holt das Programm alle 18 Zahlen aus dem Hauptspeicher, was aufgrund der relativ langsamen Rechengeschwindigkeit im Vergleich zum Speicherzugriff zu keiner Verzögerung führt. Auf einer Grafikkarte sind die Gewichte gerade andersherum verteilt. Wenn möglich, sollte daher der Zugriff auf den Hauptspeicher vermieden oder zumindest die Datenmenge reduziert werden. Dafür kann man leicht in Kauf nehmen, dass die komplette Information erst mit den aus dem Speicher geholten Daten neu berechnet werden muss. Für unser Projekt hat es sich als vorteilhaft erwiesen, nur die ersten beiden Zeilen der unitären 3x3 Matrix des Gluonfelds aus dem Speicher zu holen und danach die dritte Zeile aus den beiden ersten zu rekonstruieren. Die Reduktion der Datenmenge, die aus dem Hauptspeicher geholt werden muss (12 statt 18 reelle Zahlen) führt dabei zu einem erheblichen Netto-Gewinn an Rechengeschwindigkeit für den Wilson-Dirac-Operator.

Die hohe nominelle Rechenleistung von Grafikkarten wird ganz wesentlich dadurch erreicht, dass einzelne Programmteile (sogenannte „threads"), die unabhängig voneinander ablaufen können, auf verschiedene Prozessorkerne verteilt und parallel abgearbeitet werden. Bei der Programmierung des Wilson-Dirac-Operators haben wir versucht, jedem Gitterpunkt einen solchen Thread zuzuordnen und die Zahl der Instruktionen, die unabhängig vom benachbarten Gitterpunkt ablaufen können, zu maximieren.

Das Ergebnis unserer Bemühungen lässt sich in Abbildung 2 ablesen. Hier wird die Rechengeschwindigkeit für den Wilson-Dirac-Operator auf einer herkömmlichen CPU (Intel Core2Duo, 2,13 GHz) mit zwei verschiedenen Grafikkarten verglichen, der Nvidia 8800GT und der Nvidia GTX280. Dabei wird auch untersucht, wie sich die Gittergröße auf die Rechengeschwindigkeit auswirkt. Während die Intel-CPU eine Rechenleistung zwischen 3 und 4 GFlop/s erreicht, schaffen Grafikprozessoren die beeindruckende Spitzenleistung von mehr als 60 GFlop/s. Interessant ist dabei auch, dass GPUs ihre enorme Leistung vorwiegend auf größeren Gittern ausspielen, denn für die kleinste betrachtete Systemgröße von 4^4 liegt das Feld wesentlich dichter beisammen. Dieses Verhalten

lässt sich dadurch verstehen, dass bei kleinen Gittern die Rechenleistung durch den Speicherzugriff dominiert wird, der auf den GPUs relativ ineffizient ist.

Grafikprozessoren können also tatsächlich für extrem rechenzeitintensive wissenschaftliche Anwendungen eingesetzt werden. Allerdings ist nicht zu erwarten, dass Universitätsrechenzentren künftig vorwiegend mit Systemen aus GPUs bestückt werden, denn trotz der beeindruckenden Rechengeschwindigkeit ist der Programmieraufwand teilweise erheblich. Die Gitter-QCD mit ihren einfach strukturierten mathematischen Objekten ist vermutlich einer Programmierung auf GPUs besonders zugänglich. Für Anwendungen in der Klima- und Materialforschung mag das anders sein.

Obwohl man vom Rechenbedarf der Unterhaltungsindustrie profitiert, wird die Gitter-QCD durch den Einsatz von hochgezüchteten Grafikprozessoren nicht unbedingt zum Kinderspiel. Denn die hier gezeigten Ergebnisse sind alle auf einem einzelnen System ohne Parallelisierung realisiert worden. Das heißt, dass ein Datentransfer über Prozessorgrenzen hinweg, wie er auf dem „Wilson"-Cluster stattfindet, nicht berücksichtigt wurde. Kommerzielle Netzwerke sind aber noch zu langsam, um mit der hohen Rechengeschwindigkeit von GPUs mithalten zu können. Nachteilig ist auch, dass viele GPUs bislang nur mit 32-Bit-Genauigkeit rechnen können, was zwar für die meisten Grafikanwendungen völlig genügt (es reicht, wenn man den Gegner im Ballerspiel nur schemenhaft erkennt), für wissenschaftliche Zwecke aber nicht ausreicht. Ein vielversprechender Ansatz für die Zukunft ist daher die Entwicklung eines Systems aus einer Kombination von Grafik- und herkömmlichen Prozessoren, das die jeweiligen Vorteile der Hardware-Komponenten vereint. Solche integrierten Systeme werden bereits von einigen Herstellern angeboten.

Summary

Computer animation and video games are a hugely lucrative market worldwide. The ever increasing demand for more sophisticated visualization has driven the development of specialized graphics hardware which achieves processing speeds far exceeding those of "ordinary" processors found in workstations and servers. The article describes how the enormous computing power of graphics processors can be exploited for scientific applications. Here we focus on numerical simulations of the strong nuclear force via a technique known as lattice QCD.

Peter Pulkowski

Univ.-Prof. Dr. Hartmut Wittig

Hartmut Wittig, geboren 1963, studierte von 1982 bis 1985 Chemie in Mainz und von 1985 bis 1989 Physik in Mainz und Oxford. Die Promotion erfolgte 1992 an der Universität Hamburg, wo er 1998 auch habilitiert wurde. Er absolvierte Forschungsaufenthalte in Southampton und am Deutschen Elektronen-Synchrotron (DESY) in Zeuthen, bevor er von 1996 bis 2000 als „Advanced Fellow" an der Universität Oxford tätig war. Bis zu seiner Berufung nach Mainz im Jahr 2005 auf eine Professur für theoretische Kernphysik war er permanenter wissenschaftlicher Mitarbeiter bei DESY in Hamburg.

Kontakt

Univ.-Prof. Dr. Hartmut Wittig
Institut für Kernphysik
Johannes Gutenberg-Universität Mainz
D-55099 Mainz
Tel. +49 (0) 6131-39 26 808
Email: wittig@kph.uni-mainz.de

Literatur

1. Wilson KG. Confinement of Quarks. Phys Rev 1974; D10: 2445
2. http://wwwkph.kph.uni-mainz.de/T/644.php