Schnelle, Effiziente und Energieeffiziente Algorithmen für moderne Prozessoren und Grafikkarten in der Hochenergiephysik und anderen Gebieten[1]

David Rohr[2]

Abstract: Große Rechenzentren und Supercomputer bilden Grundvoraussetzungen vieler rechenintensiver Gebiete aus Forschung und Industrie. Sie ermöglichen das schnellere Gewinnen besserer, genauerer und umfangreicherer Resultate sowie den effizienteren Einsatz der begrenzten Ressourcen. Bisher wurde der stetige Geschwindigkeitszuwachs unter anderem durch erhöhte Energieaufnahme erkauft. Heutige Rechenzentren erreichen mit einem Strombedarf von teils weit über zehn Megawatt den Punkt, an dem höhere Rechenleistung nur noch durch effizientere Hardware und Programmierung möglich ist. Diese Arbeit stellt in drei völlig unterschiedlichen Fachrichtungen (Datenanalyse in der Hochenergiephysik am CERN, Fehlertolerante Datenspeicherung und Lineare Algebra) Lösungen zu rechenintensiven, grundlegenden Problemen vor. Diese arbeiten durch effiziente Programmierung und Einsatz moderner paralleler Prozessoren sowie Beschleunigerkarten deutlich schneller und energieeffizienter als herkömmliche Methoden

Motivation

Supercomputer haben in den letzten Jahren in vielen Bereichen große Bedeutung erlangt und ermöglichen vormals unerwartete Durchbrüche in der Forschung. Darunter fallen meteorologische Simulationen zur Wettervorhersage und Klimaforschung, Grundlagenforschung in der Hochenergiephysik oder theoretischen Physik, Quantenchromodynamik, Risikoanalysen der Finanzmärkte, Proteinfaltung, Materialforschung, Unfallsimulationen zum besseren Schutz von PKW Insassen, Neurowissenschaften und Strömungsdynamik in der Autoindustrie, Luft und Raumfahrt. Computer sind ein essentieller Faktor in diesen Feldern, die viele Petaflopjahre an Rechenleistung benötigen. Seit den siebziger Jahren folgte die Entwicklung integrierter Schaltkreise, der Grundbausteine aus welchen Computer aufgebaut sind, dem von Gordon Moore formulierten Gesetz, welches besagt, dass sich die Anzahl der Transistoren und damit grob gesagt auch die Leistungsfähigkeit von Computern etwa alle achtzehn Monate verdoppelt. Mittlerweile hat die Wärmeentwicklung der Prozessoren eine Größenordnung erreicht, die kaum mehr mit verhältnismäßigem Aufwand zu beherrschen ist. Daher begann um die Jahrtausendwende die Taktrate neuer Prozessoren zu stagnieren und blieb über die letzten Jahre quasi gleich. Der rasch wachsende Energiebedarf hat sich nicht nur für Computerprozessoren zu einem Problem entwickelt, auch Rechenzentren mit Supercomputern haben einen immensen Energiebedarf entwickelt. Seit langer Zeit wurde die stetig anwachsende Leistung der Rechenzentren unter

[1] Englischer Titel der Dissertation: “On Development, Feasibility, and Limits of Highly Efficient CPU and GPU Programs in Several Fields”

[2] Frankfurt Institute for Advanced Studies, Goethe Universität Frankfurt, Ruth-Moufang-Str, 1, 60438 Frankfurt, drohr@cern.ch

anderem durch eine erhöhte Leistungsaufnahme erkauft, deren Stromkosten die Betreiber heute kaum zu tragen bereit sind. Derzeitige Rechenzentren verbrauchen zum Teil deutlich mehr als zehn Megawatt an elektrischer Leistung, wodurch die Betriebskosten für wenige Jahre bereits die Anschaffungskosten übersteigen, Es ist eine gesellschaftliche Verantwortung die vorhandenen Ressourcen so um- und weitsichtig wie möglich einzusetzen. Insgesamt gesehen ist damit ein weiter steigender Strombedarf technisch in Bezug auf Bereitstellung und Kühlung nur aufwendig zu realisieren sowie unter ökonomischen und ökologischen Gesichtspunkten nicht akzeptabel, sowohl auf der kleinen Skala der Prozessoren als auch auf der großen Skala der Rechenzentren. Dennoch gehen die Anwender natürlich weiter von der gewohnten Steigerung der Rechenleistung aus. Aufgrund der Anschaffungskosten von vielen Millionen Euro für neue Supercomputer spielt die Kosteneffizienz eine gewichtige Rolle. Es werden Computer benötigt, die eine möglichst hohe Rechenleistung pro Prozessor bzw. pro ausgegebenem Euro bieten. Dies sind nicht notwendigerweise klassische Prozessoren, solche sind für allgemeine Programme und oft auch zur einfachen Programmierbarkeit konzipiert. Natürlich reduzieren energieeffizientere Computer bei gleicher Leistung die Stromkosten und damit auch die **T**otal **C**ost of **O**wnership (**TCO**), verbessern also die Kosteneffizienz. Ein weiterer Faktor der Kosteneffizienz besteht in der Frage, wie effizient die jeweilige Anwendung einen Prozessor nutzt. Prozessoren können in der Regel mehrere Rechenoperationen pro Takt durchführen, aber unglücklicherweise sind viele Anwendungen so implementiert, dass sie zu einer insuffizienten Nutzung der Prozessorarchitektur führen. Daher sind sowohl effiziente Prozessoren als auch effiziente Anwendungsprogrammierung erforderlich.

Diese Überlegungen führen Hardwarehersteller aus drei Gründen dazu, ihre Chips auf andere Weise als durch erhöhte Taktfrequenz zu beschleunigen: Erstens muss die Energieeffizienz von Computern erhöht werden, eine Taktsteigerung hätte aber eher den gegenteiligen Effekt. Zweitens sind höhere Takte aus technischer Sicht mit Blick auf die Kühlung kaum möglich. Und drittens müssen die Chips deutlich schneller werden, um neben der Energieeffizienz auch die Kosteneffizienz zu steigern. Die von den Herstellern dafür vorgenommenen Verbesserungen umfassen zum einen die Steigerung der pro Takt verrichteten Arbeit, was aber nur in relativ geringem Umfang möglich ist. Der wichtigere Aspekt ist der Umstieg auf breit angelegte parallele Architekturen, die in den letzten Jahren eine gewaltige Verbreitung erlangt haben. Neue Prozessoren verfügen über mehrere Kerne und Vektorinstruktionen. Leistungsfähiger und energieeffizienter als Prozessoren herkömmlicher Bauart sind Many-Core Prozessoren wie Grafikkarten (**GPUs**), die mit sehr vielen parallel arbeitenden einfach gestrickten Kernen Hunderte von Aufgaben gleichzeitig bearbeiten, oder auch Prozessoren mit breiten Vektorregistern wie beispielsweise Intels Xeon Phi. Diese parallel arbeitenden Prozessoren mit unzähligen unabhängigen, aber einfachen Rechenwerken sind konstruktionsbedingt für seriell geartete Anwendungen ungeeignet. Im Gegenzug sind sie durch ihre enorme Gesamtrechenleistung prädestiniert für parallele Aufgaben, während der simple Aufbau der Kerne eine unschlagbare Energieeffizienz garantiert. Leider können konventionelle Programme diese Vorzüge nicht nutzen. Meist reicht hierfür auch keine Neukompilierung aus, sondern die verwendeten Algorithmen müssen unter Umständen angepasst und auf parallele Weise neu implementiert werden. Die Anpassung erfordert häufig ein Durchdenken und Modifizieren des Algorithmus, um die mögliche Parallelität auszunutzen. Dies ist kompliziert, zeitaufwendig und die Rentabi-

lität schwer abschätzbar. Viele Programmierer scheuen neuartige Prozessoren und etliche Stimmen raten von deren Nutzung ab, weil sie schwer und nicht effizient programmierbar seien und dies ihre Vorteile nivelliere. Es stellt sich die Frage, welche Vorteile diese neuen Architekturen für die Forschung bieten.

Häufig lassen sich Anwendungen in serielle und parallele Unterroutinen aufteilen, wobei erstere schwer, gar nicht oder nur teilweise parallelisierbar sind. Daher sind viele moderne Cluster heterogen aufgebaut, mit breiten parallelen Prozessoren für parallele und schnellen für serielle Aufgaben. Geschickte Programmierung kann oft die langsamen seriellen Teile mit den parallelen überlappen und so deren Ausführungszeit verstecken. In jedem Fall sind jedoch hocheffiziente Programme obligatorisch, um nicht einen Großteil der Rechenleistung zu verschenken. Für die Forschung Ergeben sich damit die folgenden Aufgaben:

- Supercomputer müssen möglichst energie- und kosteneffizeint designt werden.
- Es muss untersucht werden, ob bzw. wie gut sich neue parallele Prozessortypen wie z. B. GPUs für die gegebenen Anwendungen eignen.
- Algorithmen und Programme müssen die höchstmögliche Effizienz erzielen, d. h. einerseits dass der Algorithmus dahingegen optimiert sein muss, dass er die Aufgabe mit möglichst wenig Rechenschritten erledigt und andererseits dass der Algorithmus so implementiert sein muss, dass er die gegebenen Ressourcen möglichst effizient nutzt, also pro Taktzyklus möglichst viele Rechenoperationen durchführt.

Diese Arbeit setzt sich mit drei Themen auseinander. Erstens: mit der Entwicklung schneller Algorithmen und deren Implementierung auf modernen Prozessoren und Grafikkarten. Zweitens: mit der maximal erzielbaren Effizienz in Bezug auf die spezifizierte Maximalrechenleistung (d. h. welchen Anteil der vom Computer bereitgestellten Rechenleistung kann die Anwendung ausschöpfen) und die Leistungsaufnahme (wie viel Strom benötigt die Anwendung pro durchgeführte Rechenoperation). Sowie drittens: mit der Umsetzbarkeit und den Grenzen von Programmen auf modernen Prozessoren, Grafikkarten und heterogenen Systemen. Zu diesem Zweck werden drei völlig unterschiedliche Lösungen zu grundlegenden Problemen aus verschiedenen Gebieten, die im Umfang der Arbeit realisiert wurden, vorgestellt, analysiert und verglichen. Diese folgenden drei Programme stellen jeweils für sich gesehen Lösungen für wichtige Probleme dar, und zeigen im Gesamten die Möglichkeiten paralleler und effizienter Computer.

Ereignisrekonstruktion für ALICE

Der **L**arge **H**adron **C**ollider (**LHC**) der Europäischen Organisation für Kernforschung (**CERN**) in Genf ist momentan der weltweit leistungsstärkste Teilchenbeschleuniger. **A** **L**arge **I**on **C**ollider **E**xperiment (**ALICE**) ist eines der vier großen Experimente, die am LHC installiert sind, und dient hauptsächlich dem Studium von Schwerionenkollisionen. Hierzu beschleunigt der LHC Kerne von Bleiatomen nahezu auf Lichtgeschwindigkeit und schießt diese mit einer Rate von mehreren tausend Hertz aufeinander. Eine der größten Herausforderungen bei der Auswertung liegt in der Rekonstruktion der Trajektorien dieser abertausender Teilchen, die bei jedem Zusammenstoß entstehen (**Tracking**). ALICE

besitzt mehrere Detektoren zur Spurerkennung, der wichtigste ist die **T**ime **P**rojection **C**hamber (**TPC**): eine mit Gas gefüllter Kammer. Die durchfliegenden Teilchen ionisieren Gasmoleküle, wobei man die Ionisierungspunkte (**Hits**) messen kann. Die Messung der TPC für ein Ereignis besteht aus einem riesigen Datensatz dreidimensionaler Raumpunkte, durch die jeweils eines der Teilchen geflogen ist. Die Aufgabe der Spurrekonstruktion besteht darin, aus der gewaltigen Menge dieser gemessener Raumpunkte (Hits) die Teilchenspuren (**Tracks**) zu rekonstruieren, die Messpunkte den einzelnen Teilchen zuzuordnen und die physikalischen Teilcheneigenschaften anhand der Flugbahn zu bestimmen.

Typische zentrale Blei-Ereignisse sind mehrere zehn Megabyte groß und werden mit einer Rate von einigen hundert Hz gemessen, was einer eingehenden Datenrate von bis zu $30\,^{\mathrm{GB}}/_{\mathrm{s}}$ entspricht, rechentechnisch eine enorme Herausforderung. Der ALICE **H**igh **L**evel **T**rigger (**HLT**) ist eine Rechenfarm aus circa 250 Computern, die in Echtzeit einen Großteil der Ereignisrekonstruktion durchführt. Die Aufgabe des HLT besteht in einer schnellen Analyse der Daten, einer Entscheidung welche Daten gespeichert werden und einer Kompression ebendieser Daten. Die gespeicherten Daten werden später von der ALICE Offline Software ausgewertet, die genauere und mehr Analysen liefert, aber dafür wesentlich mehr Zeit in Anspruch nimmt. Zur Echtzeitberechnung dieser Flugbahnen wurde der vom ALICE High Level Trigger (**HLT**) für die Rekonstruktion eingesetzte Algorithmus auf GPUs portiert (**GPU Tracker**) und auf 180 zu diesem Zweck mit NVIDIA Fermi Karten ausgestatteten Rechenknoten des HLT installiert.

Der Algorithmus funktioniert im Grunde intern wie folgt: Zuerst werden mit einer Heuristik auf lokaler Ebene kombinatorisch **Seeds** gesucht: sehr kurze Spurkandidaten bestehend aus wenigen Hits. Anhand des bekannten Verhaltens geladener Teilchen in einem Magnetfeld, werden im nächsten Schritt die wahrscheinlichsten Spurparameter bestimmt. Diese umfassen die Flugbahn des Teilchens sowie einige physikalische Eigenschaften wie die Ladung. Mittels der Spurparameter wird die Flugbahn des Teilchens durch die gesamte TPC extrapoliert. Darauf aufbauend werden weitere Hits gesucht, die nahe der Flugbahn liegen und vermutlich zu dieser Trajektorie gehören, der Spur zugeordnet und zur Verbesserung der Abschätzung der Spurparameter genutzt. Der Tracking-Algorithmus benutzt den **Kalman Filter** zur Extrapolation der Spuren und zur Errechnung der Spurparameter.

Einige für die Arbeit am Tracking-Algorithmus vorgenommene Optimierungen verkürzen die Bearbeitungszeit auf der GPU deutlich, verbessern die physikalischen Resultate und stellen die Reproduzierbarkeit der Ergebnisse bei nebenläufiger Ausführung des Algorithmus sicher. Beispielsweise nutzt das Programm mehrere Prozessorkerne, um die Grafikkarte hinreichend mit Daten zu füttern, und baut auf eine Pipeline mit asynchronem Datentransfer, um eine dauerhafte Auslastung der GPU auch während der Datentransfers zu gewährleisten. An mehreren Stellen ging der Algorithmus ursprünglich nicht vollständig deterministisch vor. Beispielsweise hängen bei einer naiven Implementation des Algorithmus einige Kriterien von der Bearbeitungsreihenfolge ab. Bei einer parallel arbeitenden Grafikkarte jedoch, die viele Arbeiten gleichzeitig verrichtet, ist die Reihenfolge nicht wohldefiniert. Daher können bei obigem nichtdeterministischem Kriterium durch unvermeidliche Schwankungen in der Bearbeitungsreihenfolge leicht verschiedene Resultate herauskommen. Diese unterschiedlichen Ergebnisse sind physikalisch alle gleichwertig,

erschweren allerdings die Qualitätssicherung ungemein, da man die Ausgabe von zwei Läufen nicht auf unterster Ebene vergleichen kann. Alle solche Effekte, die von Nebenläufigkeit herrühren, wurden eliminiert; meist durch Wahl besserer Kriterien, wodurch nebenbei sogar Effizienz und Auflösung der Spurrekonstruktion verbessert wurden.

Nach all diesen Optimierungen arbeitet der am HLT eingesetzte Tracking Algorithmus auf dem Prozessor um den Faktor 15 schneller als die "Offline"-Spurrekonstruktion, der von der ALICE Offline Gruppe zu Datenanalyse eingesetzt wird. Im Falle der auf den HLT Rechenknoten für die Spurrekonstruktion verfügbaren Ressourcen, ermöglicht die Spurrekonstruktion auf der GPU einen weiteren Geschwindigkeitszuwachs um den Faktor 10, ist also insgesamt 150 mal so schnell wie der Offline Algorithmus auf dem Prozessor. Abgesehen von der verkürzten Rechenzeit bewirkt der GPU Tracker auch eine deutliche Senkung der Kosten im HLT. Stünde der GPU Tracker nicht zur Verfügung, so müssten herkömmliche Prozessoren die Spurrekonstruktion leisten wofür deutlich mehr Rechenknoten von Nöten wären. Unter Berücksichtigung der Hardwarepreise ergibt sich nur für die seit 2012 eingesetzten Server durch den Einsatz der 180 GPUs eine Kostenersparnis von 300000 $ – zusätzlich benötigte Infrastruktur und Strom für mehr Server nicht mitgerechnet. Die Ersparnis für den neuen HLT im ALICE Run 2 ab 2015 ist noch deutlich größer und liegt im siebenstelligen Bereich.

Nach einer Testphase arbeitete die GPU-basierte Spurrekonstruktion seit Anfang 2012 fehlerfrei im Dauerbetrieb und ermöglichte die vollständige Online-Rekonstruktion der Teilchenspuren. Abgesehen vom GPU Tracker wurde auch der Track Merger, der zusammengehörige Spursegmente zusammensetzt, für GPUs umgesetzt. Hier konnte jedoch gezeigt werden, dass sich aufgrund der limitierten PCI Express Bandbreite keine deutliche Verkürzung der Bearbeitungszeit erzielen lässt. Daher ist es sinnvoller, den Tracker, der davon deutlich mehr profitiert, auf der GPU auszuführen, und den Merger auf dem Prozessor zu belassen. Es muss also fairerweise gesagt werden, dass GPUs kein Allheilmittel sind, sondern dass es Anwendungen gibt, für die GPUs prinzipiell ungeeignet sind. In diesen Fällen ist es wichtig, wie beim Track Merger geschehen, anhand einfacher Kriterien zu erkennen, ob und warum GPUs für die Aufgabe ungeeignet sind, so dass keine Arbeitszeit verschwendet wird. Oftmals ist in solchen Fällen dennoch eine effiziente Implementierung möglich, zum Beispiel durch Einsatz der Vektorinstruktionen von Prozessoren. Viele geplante neue Experimente, wie z. B. das **CBM** Experiment des **FAIR** Projekts an der GSI in Darmstadt, arbeiten mit wesentlich höheren Datenraten, wodurch eine Speicherung der Rohdaten unmöglich wird und eine vollständige Echtzeitrekonstruktion obligatorisch ist. Diese Experimente sind ohne Fortschritte bei der schnellen Rekonstruktion, wie z. B. der Spurrekonstruktion für ALICE, schlicht nicht realisierbar.

Benchmarks für Heterogene Supercomputer

Der Linpack Benchmark (oft auch **H**igh **P**erformance **L**inpack (HPL) genannt) ist das Standardwerkzeug zur Klassifikation der Rechenleistung von Hochleistungsrechnern. Linpack löst ein dicht besetztes lineares Gleichungssystem iterativ mittels *LU*-Faktorisierung mit zeilenweiser Pivotisierung. Jede Iteration besteht aus mehreren Schritten. Das Gros

der Rechenzeit beansprucht dabei eine Matrix-Matrix-Multiplikation, die von einer Routine mit Namen **DGEMM** durchgeführt wird. Der **LOEWE-CSC** ist ein im Herbst 2010 installierter Supercomputer der Goethe Universität Frankfurt, bestehend aus circa 800 Servern mit GPU Beschleunigerkarten. **Sanam** ist ein Großrechner der King Abdulaziz City for Science and Technology, der in Kooperation mit dem Frankfurt Institute for Advanced Studies (**FIAS**) geplant und installiert wurde. Die Evaluation und Zusammenstellung der Hardware für Sanam wurden vollständig im Rahmen dieser Arbeit vorgenommen.

Da keine für AMD GPUs optimierten Versionen von DGEMM und HPL existierten, wurde eine DGEMM Bibliothek für GPUs mit Namen **CALDGEMM** und darauf aufbauend eine Linpack Implementierung (**HPL-GPU**) programmiert, die beide als Open Source Software frei verfügbar sind. Die Implementierung nutzt einen in GPU-Assembler geschriebenen DGEMM Kernel zur Matrixmultiplikation auf GPUs, der mehr als 90 % der theoretischen Maximalrechenleistung erreicht, und belässt die verbleibenden Schritte auf dem Prozessor, da diese auf der GPU nicht ebenso effizient implementiert werden können wie DGEMM. Für HPL-GPU wurde ein neuartiges, auf mit GPU-Beschleunigern ausgestattete Computer zugeschnittenes Feature entwickelt, das die seriellen Teile des HPL hinter der GPU Berechnung versteckt. Dieses Feature, eine Pipeline und asynchroner Datentransfer ermöglichen dem Prozessor die ihm zugewiesenen Aufgaben parallel zur Matrixmultiplikation auf der Grafikkarte zu verrichten und stellen damit die vollständige Auslastung beider Chips sicher. Mittlerweile hat Intel die hier erwähnten Verbesserungen weitestgehend in deren HPL Implementierung übernommen. Um mit vielen Hardwarearchitekturen kompatibel zu sein, kann CALDGEMM wahlweise die Schnittstellen **CAL**, **CUDA** und **OpenCL** zur GPU-Programmierung verwenden.

Der LOEWE-CSC Cluster benutzt verschiedene Rechenknoten für verschiedene Anwendungen: eine Sorte mit mehr CPU Kernen und mehr Speicher und eine Sorte mit GPUs. Herkömmliche Linpack-Implementierungen verteilen die Rechenlast völlig gleichmäßig auf alle vorhandenen Knoten und können mit solch einer heterogenen Situation nicht umgehen, denn sie bremsen damit alle Knoten auf das Niveau des langsamsten teilnehmenden Knotens herunter. HPL-GPU kann die Matrix der vorhandenen Rechenleistung entsprechend verteilen. Hierfür musste insbesondere der Algorithmus zum Lösen von Dreiecksystemen modifiziert werden. Ein exemplarischer Test mit sechs Knoten drei verschiedener Leistungsklassen erreichte 96,9 % der akkumulierten Einzelleistungen aller Knoten und damit lediglich 3,1 % Granularitätsverlust.

Da GPUs nur sehr einfache, aber dafür viele Kerne einsetzen, sind sie für optimierte Anwendungen konstruktionsbedingt energieeffizienter als Prozessoren. Beim HPL ist es möglich, die Energieeffizienz weiter zu steigern, indem man möglichst viele Teile des Algorithmus von der CPU auf die GPU auslagert und die Prozessoren damit soweit als möglich absichtlich brach liegen lässt. HPL-GPU bietet einen effizienzoptimierten Modus, der auf dem Sanam zwar eine um 11,1 % reduzierte Leistung, dafür aber eine um 23,2 % reduzierte Stromaufnahme ermöglicht. Die Energieeffizienzsteigerung steigt um 15,8 %. Auch für spezielle Systeme im Niedrigenergiebereich gibt es besondere Anpassungen, um deren verhältnismäßig langsame Prozessoren zu kompensieren. Mit optimierten Versionen von HPL-GPU konnte bereits Anfang 2011 sowohl auf einem solchen Niedrigener-

giesystem von **SDS** als auch auf einem multi-GPU System die ersten Messungen einer Energieeffizienz von mehr als $1\,\mathrm{GFlop/J}$ mit einem GPU System demonstriert werden.

Unter Einsatz von CALDGEMM und HPL-GPU platzierte sich der LOEWE-CSC im November 2010 auf Platz 22 in der Top500-Liste der weltweit leistungsstärksten Supercomputer. Sanam erreichte im November 2012 den zweiten Platz in der Green500-Liste der energieeffizientesten Supercomputer und musste sich nur einem von Intel zu diesem Zweck geförderten viel kleineren System geschlagen geben. HPL-GPU bot damit auf dem LOEWE-CSC eine bis dato auf GPU-Clustern unerreichte Effizienz im Vergleich zur theoretischen Maximalleistung und erst kürzlich konnten andere Implementierungen von NVIDIA und Intel aufschließen, wobei NVIDIA z. B. auf dem Titan Supercomputer nur auf den Grafikkarten rechnet und die Prozessoren außen vorlässt, und Intel den Ansatz der Arbeitsverteilung auf Prozessor und GPU von HPL-GPU übernommen hat.

Basierend auf diese Arbeit wurde im Herbst 2014 an der GSI in Darmstadt der neue **Lattice-CSC** Computer für Simulationen im Feld der Quantenchromodynamik gebaut, der in gewisser Weise architekturell der Nachfolger von Sanam ist. Ein einziger Computerknoten von Lattice-CSC erreicht mit CALDGEMM eine Matrix-Multiplikationsperformance von ca. $8\,\mathrm{TFlop/s}$. Der Cluster Lattice-CSC erzielte mit HPL-GPU eine Energieeffizienz von $5,37\,\mathrm{GFlop/J}$, also 5,37 Milliarden Fließkomma-Rechenoperationen in doppelter Genauigkeit pro Sekunde und pro Watt elektrischer Leistung. Damit erlangte Lattice-CSC den ersten Platz der Green500 Liste und ist momentan der energieeffizienteste Supercomputer der Welt. Mit Baukosten von 1,7 Millionen Euro stellt Lattice-CSC eine theoretische Maximalrechenleistung von $1,7\,\mathrm{PFlop/s}$ bereit. Damit liegt er auch in Bezug auf die Kosteneffizienz vor fast allen anderen Supercomputern.

Fehlertolerante Kodierung

Die Menge der von der Menschheit produzierten digitalen Daten steigt rasant an und hat inzwischen mit vielen Exabyte pro Jahr eine unvorstellbare Dimension erreicht. Dies stellt eine enorme Herausforderung bei der Datenspeicherung dar. Unzählige Speichermedien müssen parallel eingesetzt werden, wodurch das regelmäßige Versagen einzelner Medien vorprogrammiert ist. Bei sehr großen Datenzentren kann dies den Ausfall mehrerer Medien pro Tag bedeuten. Die Entwicklung von Mechanismen zur Vermeidung von Datenverlusten ist evident. Die Kodierungstheorie ermöglicht die einfache, fehlertolerante Datenspeicherung und ist damit eine absolute Notwendigkeit für jedwede heutige Computerinfrastruktur. Sie verhindert Datenverluste, Übertragungsfehler und Systemabstürze bei Hardwaredefekten. Die Mathematik hinter vielen fehlertoleranten Codes findet in endlichen Körpern ($\mathbb{F}_q$ genannt) statt. Im Folgenden bezeichnet ein (n,k)-Code einen Code, der n Datenwörter in $n+k$ Codewörter kodiert, so dass beim Verlust von bis zu k Codewörtern alle Datenwörter wiederhergestellt werden können. Praktisch heißt das, man benötigt $n+k$ identische Speichermedien (z. B. Festplatten), hat aber nur die Gesamtkapazität von n Medien zur Verfügung, dafür können dann bis zu k Medien ausfallen. Völlig unabhängig davon welche Medien ausfallen, ist in diesem Fall immer eine vollständige Rekonstruktion aller Daten möglich, so lange es nicht mehr als k sind. Prominente Beispiele

solcher Codes sind **Reed-Solomon Codes**, deren Funktionsweise sich auf Eigenschaften der Vandermonde-Matrix stützen, oder darauf aufbauende **Cauchy-Reed-Solomon Codes**. Die eigentliche Kodierung bzw. die Wiederherstellung von Daten aus den Codewörtern wird über eine Matrix-Vektor-Multiplikation realisiert. In den obigen zwei Beispielen liegen die Werte in den mathematischen Körpern $\mathbb{F}_{2^{32}}$ bzw. in $\mathbb{F}_2$. Das Problem hierbei ist, dass die Rechenoperationen in diesen Körpern von Computern nicht nativ unterstützt werden. Es gibt keine Prozessor-Instruktion, die eine entsprechende Berechnung durchführt, sondern es müssen viele Instruktionen, oft in der Größenordnung von einhundert, abgearbeitet werden, um eine einzige Rechenoperation zu emulieren. Damit sind diese Codes leider nicht für die Ausführung auf Computern geeignet.

Diese Arbeit behandelt die Kodierung in zwei Schritten. Ein erster theoretischer Teil diskutiert Codes, deren Operationen sich besser auf die vorhandenen Rechenoperationen von Computern abbilden. Es wird eine Methode vorgestellt, mit Hilfe der Ganzheitsringe algebraischer Zahlkörper Codes basierend auf Integer-Rechenoperationen in $\mathbb{Z}/2^{32}\mathbb{Z}$ zu erzeugen. Weiter wird eine neue Herangehensweise präsentiert, welche die Generierung der Kodiermatritzen beschleunigt. Außerdem können Cauchy-Reed-Solomon Codes durch geeignete Vektorisierung so formuliert werden, dass die Rechenoperationen in $\mathbb{F}_2^{32}$ durchgeführt werden können. Rechnungen in diesen beiden Objekten können Computer effizient bewerkstelligen, denn sie bilden sich direkt auf vorhandene Computerinstruktionen ab. Das Umschreiben der Kodierung von Matrix-Vektor-Multiplikation, die per Definition bandbreitenlimitiert ist, auf Matrix-Matrix-Multiplikation stellt die volle vorhandene Rechenleistung für die Kodierung zur Verfügung und ermöglicht es, insbesondere die bei der Arbeit am Linpack Benchmark erworbene DGEMM Expertise anzuwenden.

Der zweite Teil stellt neue schnelle Implementationen der besprochenen Codes vor, die in Form der Open Source Bibliothek **QEnc** frei zur Verfügung gestellt werden. Hierfür werden zuerst vorhandene leistungsfähige DGEMM Bibliotheken zur Fließkomma-Matrixmultiplikation abgewandelt, um eine passende Implementierung für QEnc zu erhalten. Hierfür werden FLießkomma-Operationen durch Berechnungen in den oben erwähnten Objekten $\mathbb{Z}/2^{32}\mathbb{Z}$ und $\mathbb{F}_2^{32}$ ersetzt. Für relativ große Matrizen ($n,k \geq 48$) erreichen diese Bibliotheken die spezifizierte Spitzenleistung der Prozessoren. Die Cauchy-Reed-Solomon Codes erlauben noch eine weitere Optimierung, die jedoch für Computer gewöhnlich nicht allgemein für beliebige Parameter n und k gleichzeitig realisiert werden kann, sondern nur für fest vorgegebene Werte von n und k. Hierbei wird ausgenutzt, dass Multiplikationen in $\mathbb{F}_2$ trivial sind. Die einzigen möglichen Faktoren sind 0 und 1. Ist der Faktor 1 wird die Multiplikation übersprungen, ist der Faktor 0 wird sowohl die Multiplikation als auch die darauf folgende Addition übersprungen. Diese Methode kann im Mittel 75 % der Rechenoperationen einsparen, setzt aber eine a priori Kenntnis der jeweiligen Kodiermatrix voraus, die von den Parametern n und k abhängt.

QEnc löst dieses Problem durch Generieren und Ausführen von binärem Assemblercode zur Laufzeit (**Just in Time** Compilation/Assembly), so dass die spezifische Matrix für die aktuellen Kodierparameter berücksichtigt werden kann. Verglichen mit der ersten naiven Implementierung, bei der die schnellsten verfügbaren DGEMM-Bibliotheken eingesetzt wurden, kann QEnc die Matrix-Multiplikation $9,1$ mal so schnell durchführen. Der op-

timierte Assembler Code erreicht durch Instruction-Level-Parallelism eine Ausführungsrate von 2,61 XOR-Rechenoperationen pro Takt. QEnc kann mittels OpenMP mehrere CPU-Kerne gleichzeitig nutzen. Auf Prozessoren kodiert QEnc mit über $10\,\mathrm{GB/s}$ bei Matrixgrößen bis etwa $n \cdot k < 400$, was mehrfach schneller ist, als andere Implementierungen selbst bei kleinen Matrizen schaffen. Die Limitierung auf $10\,\mathrm{GB/s}$ rührt nicht von der eigentlichen Rechengeschwindigkeit her, sondern der Hauptspeicher des Computers kann die Eingangsdaten schlicht nicht schneller liefern bzw. die Ausgangsdaten nicht schneller wegschreiben. Durch Generierung von **OpenCL** und **VHDL** Code kann QEnc Grafikkarten und **FPGAs** zur Beschleunigung einsetzen. Bei kleinen Parametern (n, k) demonstrieren GPUs immense Bandbreiten von über $250\,\mathrm{GB/s}$ und FPGAs können auf dem Chip eine Kodierbandbreite von mehreren Terrabyte pro Sekunde erreichen und spielen damit in einer völlig anderen Liga. Leider steht diese enorme Rechenleistung wegen der begrenzten PCI Express Bandbreite nicht für reale Anwendungen zur Verfügung.

Die Implementierung stößt für jedwede Parameter an harte Grenzen der eingesetzten Hardware. Entweder die für das Lesen der Rohdaten und Speichern der kodierten Daten benötigte Speicherbandbreite übersteigt bereits die Kapazität des Speichercontrollers. Dies bremst dann den Prozessor zwangsläufig aus. Ansonsten werden über 90 % des unter Berücksichtigung von Instruction Level Parallelism möglichen maximalen Instruktionsdurchsatzes an Vektorbefehlen erreicht. Die Hardware wird also in jedem Fall optimal genutzt. Das Kodierungsproblem kann damit für die oben angegebenen Parameter (die für fast alle Fälle ausreichend sind) als gelöst angesehen werden, da aufgrund der begrenzten Geschwindigkeit mit der die zu kodierenden Daten aus dem Speicher geladen werden können, schlicht keine weitere Beschleunigung möglich ist. Bei größeren Parametern ist in der Tat die Rechengeschwindigkeit der limitierende Faktor, aber auch hier gibt es nach Wissen des Autors keine schnellere Kodierimplementierung als in QEnc.

Vergleich der Ergebnisse & Schlussfolgerung

Zum zielgerichteten Vergleich von CPU und GPU dient die Metrik γ, welche über das Verhältnis der Effizienz auf der GPU zur Effizienz auf der CPU jeweils in Bezug zur theoretischen Maximalleistung definiert ist. Mit a_g und a_c den auf GPU und CPU erreichten Leistungen und p_g und p_c den theoretischen Spitzenwerten definiert sich γ durch:

$$\gamma = \frac{a_g/p_g}{a_c/p_c} = \frac{a_g/a_c}{p_g/p_c}.$$

Die Metrik vergleicht nicht den Leistungsunterschied zwischen GPU und CPU. Dies ist nicht zweckdienlich, da die GPU konstruktionsbedingt der schnellere Prozessor ist. Vielmehr setzt die Metrik den Leistungszuwachs in Bezug zur Spitzenperformance. Die Anwendung dieser Metrik auf die hier vorgestellten Programme ergibt in fast allen Fällen einen Wert zwischen 0,7 und 1,0, d. h. dass die GPU Implementierungen ähnlich effizient arbeiten wie ihre CPU Analoga. Bei den Beispielen mit $\gamma < 0{,}7$ konnten immer eindeutige Flaschenhälse identifiziert werden, wie z. B. die PCI Express Bandbreite beim Track Merger, die eine sinnvolle Nutzung der GPU verhindern.

Es wurden in drei verschiedenen Fachgebieten effiziente Algorithmen und Implementierungen dieser Algorithmen auf Prozessoren und Grafikkarten vorgestellt. Insgesamt konnten alle drei vorgestellten Anwendungen eine deutliche Leistungs- und Effizienzsteigerung erreichen. Ohne die schnelle GPU Implementierung wäre eine Echtzeit-Spurrekonstruktion bei voller Rate für das ALICE Experiment aus Kostengründen nicht möglich und neue Experimente die ausschließlich auf Echtzeitkonstruktion setzen wären nicht realisierbar. Die Linpack Implementierung hat mit dem Erreichen des Spitzenplatzes in der Green500 Liste ihre Energieeffizienz unter Beweis gestellt. Die entwickelte Routine zur Matrixmultiplikation im Linpack erreicht 90 % der theoretischen spezifizierten Maximalperformance der GPU. Die fehlertolerante Datenkodierung ist nicht länger durch die Rechengeschwindigkeit, sondern durch Laden und Wegschreiben der Rohdaten limitiert. Die erhöhten Leistungen rühren einerseits von Optimierungen am Algorithmus her und andererseits vom effizienten Einsatz geeigneter moderner paralleler Prozessortypen. Es stellt sich heraus, dass nicht jede Architektur für jedes Problem geeignet ist, dass es aber (zumindest bei den untersuchten Fällen) für jedes Problem eine geeignete Architektur gibt. In den meisten Fällen lässt sich die Eignung aber anhand einfach zu prüfender Kriterien relativ schnell abschätzen Es lässt sich folgern, dass feldübergreifend bei guter Programmierung eine effiziente Implementierung der meisten Algorithmen auf parallelen Architekturen wie z. B. GPUs möglich ist. In der Regel sind mindestens eine Pipeline und asynchroner Datentransfer zur auszuschöpfen der gebotenen Rechenleistung nötig. Der Aufwand entlohnt dafür aber mit Steigerungen der Leistung und der Energieeffizienz sowie mit einer günstigeren Total Cost of Ownership.

Literaturverzeichnis

[Ro14] Rohr, David: On Development, Feasibility, and Limits of Highly Efficient CPU and GPU Programs in Several Fields. Universität Frankfurt, 2014.

David Rohr wurde am 18.9.1983 in Mannheim geboren. Nach dem Abitur studierter er Physik und Mathematik an der Universität Heidelberg Als Physik-Diplomarbeit entwickelte er ein System zur automatischen Rekonstruktion von Teilchenspuren für das ALICE-Experiment am LHC am CERN unter Zuhilfenahme von GPUs. Thema seiner Doktorarbeit an der Goethe-Universität Frankfurt war die Fortführung dieser Arbeit für das CERN sowie andere Projekte im Bereich des Hochleistungsrechnen: insbesondere fehlertolerante Datenkodierung und effiziente Implementierungen des Linpack-Benchmarks auf GPUs. Momentan arbeitet er als Postdoc für das Frankfurt Institute for Advanced Studies. Dort war er unter Anderem verantwortlich für das Design und die Linpack-Optimierungen für den Supercomputer Lattice-CSC am GSI Helmholtzzentrum für Schwerionenforschung in Darmstadt, der im November 2014 von der Green500 Liste als energieeffizientester Computer der Welt ausgezeichnet wurde.