COMPUTATIONAL PHYSICS

# Grafikkarten für die Datenflut

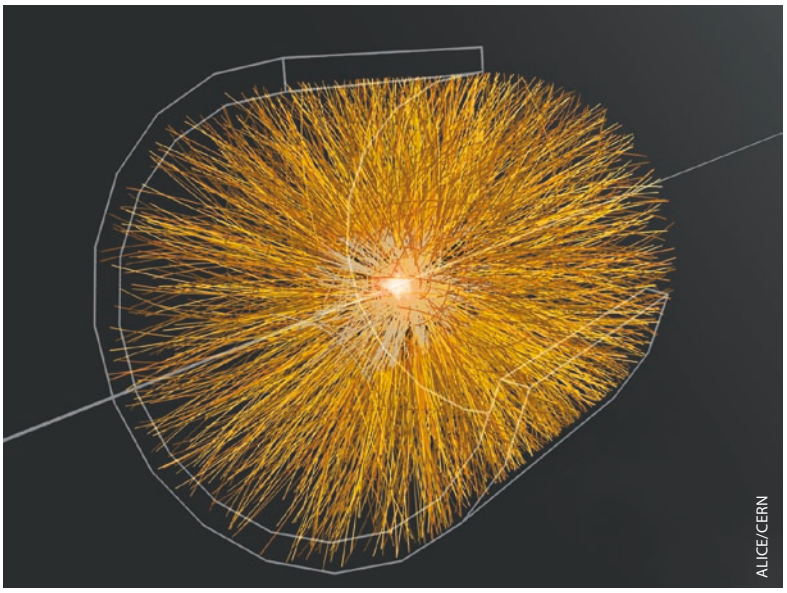
## Wie Höchstleistungscomputer mit handelsüblichen PC-Grafikkarten noch schneller rechnen

**Volker Lindenstruth**

**In vielen Bereichen der modernen Physik wächst der Bedarf an Computerleistung stetig. Oft sind mehr als eine Billiarde Rechenoperationen pro Sekunde nötig, um die Datenmassen zu erfassen und auszuwerten oder komplexe physikalische Prozesse zu simulieren. Höchstleistungscomputer verschlingen immer größere Summen und verursachen immense Betriebskosten, schon allein durch ihren Energiebedarf. Doch handelsübliche Grafikkarten, wie sie sich in PCs finden, bieten hier besonders interessante Alternativen.**

Welche Bedeutung Höchstleistungscomputer für die Forschung haben, macht die Hochenergiephysik besonders deutlich. Dort gilt es, zunehmend komplexere Detektorsysteme mit immer mehr Sensoren immer schneller auszulesen. Sowohl die Rate der aufgezeichneten Ereignisse als auch die Anzahl der zu messenden Teilchen pro Ereignis steigen. Beim Large Electron-Positron Collider (LEP), dem Vorgänger des Large Hadron Colliders (LHC) am CERN fanden in der Regel noch weniger als hundert Ereignisse pro Sekunde statt, und jedes Ereignis umfasste rund 100 Kilobyte an Daten. Die Daten ließen sich im Wesentlichen nach der Aufzeichnung auswerten. Mit jeder neuen Generation von Detektoren ist die Anforderung an die Computer-Infrastruktur um mehr als eine Größenordnung gestiegen.

Im ALICE-Experiment [1] am LHC beispielsweise sind die Raten mittlerweile so groß, dass aufwändige Selektionssysteme („Trigger“) den Datenstrom durch intelligente Auswahl der relevantesten Ereignisse reduzieren müssen [2]. ALICE dient primär dazu, Kollisionen schwerer Ionen zu untersuchen, bei denen ein sog. Quark-Gluon-Plasma entsteht. Die ersten Experimente mit Blei-Ionen haben 2010 begonnen. Der Detektor ist so konstruiert, dass er möglichst alle geladenen Teilchen nachweisen kann, die bei einer Kollision entstehen. Das sind immerhin einige tausend pro Kollision. Die Kollisionsrate im Endausbau liegt bei 8 Kilohertz und die Auslese-Datenrate beträgt mehr als 20 Terabytes pro Sekunde, die ohne Selektionssysteme nicht mehr zu bewältigen wären. Dazu ist es erforderlich, das Ereignis online zu rekonstruieren. Um die physikalisch besonders interessanten, seltenen Ereignisse selektieren zu können, muss man die bei jeder Kollision entstandenen Teilchen bestimmen, mehr als eine Million pro Sekunde. Dies geschieht anhand




Die unzähligen Teilchenspuren, die bei der Kollision von Blei-Ionen im ALICE-Experiment entstehen, lassen sich nur mit ausreichend großer Rechenleistung online rekonstruieren.

der charakteristischen Teilchenspuren, die sich durch die verschiedenen Detektoren nachweisen lassen. Bei ALICE stammen mehr als 80 Prozent der gesamten Daten vom zentralen Spurdetektor, der „Time Projection Chamber“ (TPC) (**Abb. 1**). Die im Folgenden am Beispiel der TPC erläuterten schnellen Spurrekonstruktionsalgorithmen lassen sich allgemein einsetzen und wurden schon erfolgreich auf den „Inner Tracker“ und das Myonen-Spektrometer übertragen.

Jedes der Teilchen hinterlässt auf seinem Flug durch die 88 Kubikmeter Gas der TPC eine Ionisations-

**KOMPAKT**

- In den Experimenten der Hochenergiephysik gilt es, den riesigen Datenstron durch computergestützte Selektion auf die relevantesten Ereignisse zu reduzieren.
- Die dafür nötige Rechenleistung lässt sich nur durch einen hohen Grad von Parallelität erreichen, die bereits beim Algorithmusdesign zu berücksichtigen ist.
- Die Kombination der inhärent parallelen Methode der Zellularen Automaten mit einem optimierten Kalman-Filter erlaubt einen hohen Grad an Parallelität bei gleichzeitig hoher Rekonstruktionsqualität.
- Herkömmliche Grafikkarten (Graphics Processing Unit, GPU) mit ihren mehr als tausend parallel arbeitenden Rechenwerken dienen als Rechenbeschleuniger.

**Prof. Dr. Volker Lindenstruth**, Institut für Informatik, Frankfurt Institute for Advanced Studies, Johann-Wolfgang-Goethe Universität, Ruth-Moufang-Str. 1, 60438 Frankfurt/Main

ladungsspur, die in Form vieler Raumpunkte („Hits“, **Abb. 2a**) an den beiden Endkappen des zylindrischen Volumens elektronisch nachgewiesen wird. Diese Hits verteilen sich regelmäßig entlang der Wegstrecke und bilden somit eine Art Kette entlang der Teilchenspur. Aus der rekonstruierten Spur lässt sich dann auf das Teilchen zurückschließen.

Die Teilchenspur lässt sich aus der lokalen Beziehung zwischen den Hits mit Hilfe der bekannten Methode der Zellularen Automaten bestimmen [3]: Hierbei ergeben sich eindeutige Nachbarn für jeden Hit, falls dieser genau einen Nachbarn an jeder Seite besitzt („Neighbours Finder“). Der Treffer und seine beiden Nachbarn werden dann als ein kurzes Segment einer Teilchenspur betrachtet. Solche Segmente sind leicht zu einer zusammenhängenden Spur zu verbinden, indem man ihre Nachbarn verfolgt („Tracklet Constructor“). Allerdings besteht in der Realität besonders dort, wo sich Spuren kreuzen, nicht immer eine eindeutige Beziehung zwischen nächsten Nachbarn (**Abb. 2b**). Ebenso können Spuren in mehrere Segmente zerbrechen, wenn Hits fehlen. Dennoch lässt sich eine hohe Effizienz bei der Spurrekonstruktion erreichen, wenn die Akzeptanzbedingungen für benachbarte Hits und für die Kombination der Segmente geeignet gewählt sind.

Spuren sind mehr als eine Verkettung von Raumpunkten. Die physikalisch relevanten Spurparameter wie Ort am inneren Rand der TPC, Richtung und Impuls, lassen sich mit der Kalman-Filter-Methode berechnen [4]: Dieses iterative Verfahren beginnt mit einer Schätzung der Spurparameter und verbessert sie dann durch das sukzessive Hinzufügen der einzelnen mit der Zellulare-Automaten-Methode identifizierten Hits entlang der Spur. Die genauesten Spurparameter und deren Fehler ergeben sich nach dem Verarbeiten des letzten Hits auf der Spur. An dieser Stelle kommen weitere Qualitätsfilter zum Einsatz, die falsche Spuren aussortieren.

Ein solcher hochentwickelter Algorithmus („Tracklet Selector“), der auf Zellularen Automaten und Kalman-Filtern basiert, ist fähig, schnell und effizient Ereignisse mit den sehr komplexen Spurtopologien im ALICE-Detektor zu rekonstruieren. Die typischen Kollisionen von schweren Ionen beim LHC produzieren pro Sekunde einige tausend solcher Ereignisse (**Abb. 2b**), sodass die Spurrekonstruktionsrate einige Megahertz übersteigt.

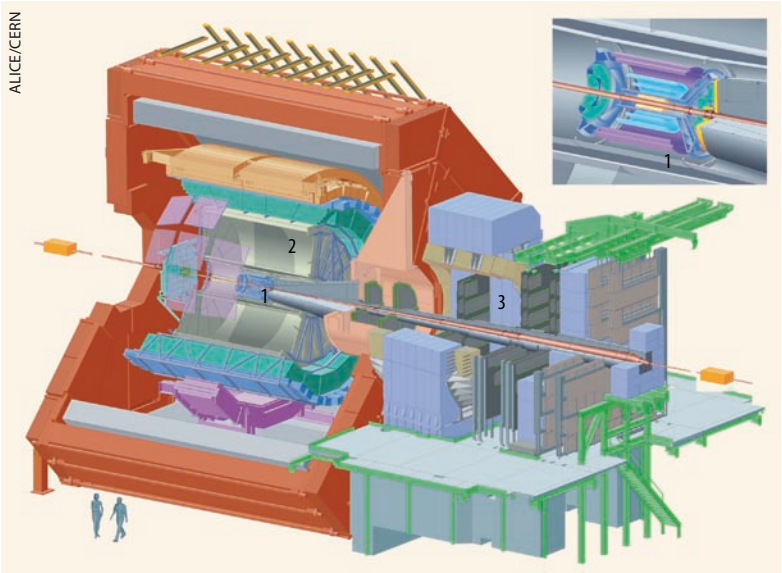



**Abb. 1** Das ALICE-Experiment besteht im Kern aus dem Inner-Tracking-System (1), das die genaue Position der Kollision bestimmt, und dem zentralen Spurdetektor, der Time Projection Chamber (TPC) (2), die alle geladenen Teilchen im Magnetfeld aufzeichnet. Die stark vorwärts gerichteten Myonen werden vom Myon-Spektrometer (rechts) nachgewiesen (3).

## Schneller durch Parallelität

Die Integrationssteigerung von Mikrochips basiert darauf, dass sich auf Halbleitern immer feinere Strukturen photolitographisch aufbringen lassen – die Grundlage von Moores Gesetz. Auf diese Weise gelingt es, immer kleinere und schnellere Transistoren im gleichen Fertigungsschritt zu produzieren. Doch dies hat aus verschiedenen physikalischen Gründen seine Grenzen erreicht. Zwar kann die Miniaturisierung noch eine Zeit lang weiter voranschreiten, doch die heute erreichbaren Taktraten lassen sich kaum weiter steigern. Höhere Rechenleistungen basieren vielmehr auf einer größeren Anzahl unabhängiger Rechenwerke. Computer werden also im Wesentlichen nicht mehr schneller, sondern paralleler. Dies hat natürlich fundamentale Auswirkungen auf die Programmierung solcher Systeme.

Seit kurzem sind Prozessoreinheiten mit zwölf kompletten Einzelprozessoren („Kernen“) verfügbar. Wegen der wachsenden Zahl unabhängiger Kerne innerhalb eines Prozessors wird es immer wichtiger, wie der Hauptspeicher angebunden ist, und wie die Algorithmen mit den Datenstrukturen im Hauptspeicher kommunizieren. Wenn etwa ein benötigtes Datenwort aus dem Hauptspeicher gelesen werden muss, weil es nicht im Zwischenspeicher („Cache“) zu finden war, entstehen Wartezeiten. Sie können derzeit von 35 bis 130 Nanosekunden reichen. Bei einer Taktrate von drei Gigahertz entspricht dies 105 bis 390 Wartezyklen oder 420 bis 1560 nicht ausgeführten Befehlen, da sich pro Takt vier oder mehr Operationen ausführen lassen. Wenn also die Gefahr eines solchen „Cache Misses“ besteht, ist es daher oft ökonomischer, einen komplexen Wert neu zu berechnen anstatt ihn aus dem Speicher zu lesen. Auch bei komplexen Tabellen ist es oft sinnvoll, zu prüfen, ob sie sich nicht durch Interpolation der Werte verkleinern lassen.

Ein weiterer für die Speicherbandbreite relevanter Punkt ist die Genauigkeit der Berechnung. Einfache Rechengenauigkeit (32 Bits) hat einen numerischen Fehler von $6 \times 10^{-8}$, doppelte Rechengenauigkeit (64 Bits) einen Fehler von $1 \times 10^{-16}$. Die wenigsten Observablen werden mit einer höheren Präzision als einfacher Genauigkeit gemessen. Algorithmen, die doppelt genaue Zahlen vermeiden, benötigen zudem im Vergleich nur halb so viel Hauptspeicher und sind dadurch wesentlich effizienter zu verarbeiten, selbst

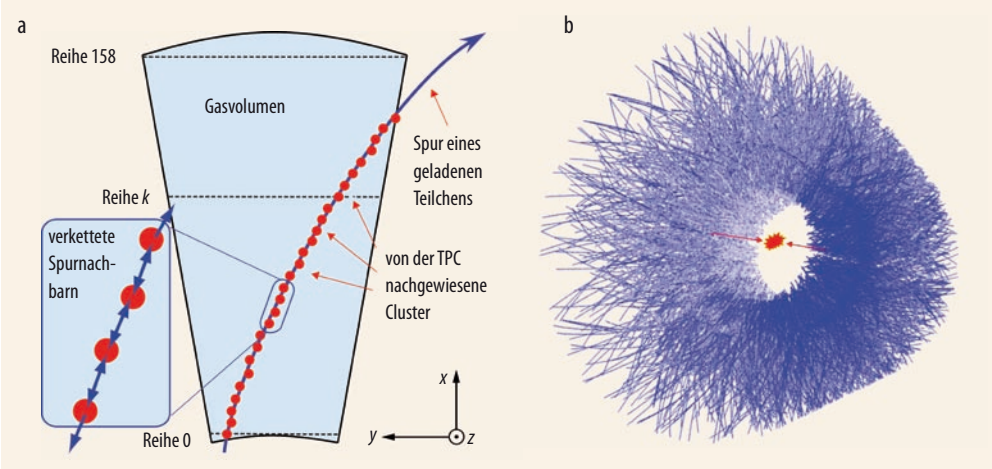



**Abb. 2** Die unzähligen einzelnen Hits müssen zu Spursegmenten und diese zu vollständigen Teilchenspuren verbunden werden (a), um schließlich das ganze Ereignis in der Detektorkammer rekonstruieren zu können (b).

wenn dies einige Korrekturschritte erfordert. Wer mit höherer Genauigkeit als nötig rechnet, produziert daher teure Zufallszahlen.

Die Bedeutung des korrekten Einsatzes der Rechengenauigkeit wächst im Kontext der Vektorisierung, wenn also ein Befehl auf mehrere Werte angewendet werden soll. Solche Instruktionen heißen „Single Instruction – Multiple Data" (SIMD). Durch die wachsende Zahl der Transistoren wächst die interne Registerbreite der Prozessoren. 128 Bit sind heute üblich, 512-Bit-Systeme sind aber bereits angekündigt. Mit nur wenig Hardware-Aufwand ist es aber möglich, statt einer 128-Bit-Operation zwei 64-Bit- oder vier 32-Bit-Operationen gleichzeitig durchzuführen (**Abb. 3**). Bei herkömmlicher Programmierung berechnen die breiten Register nur eine Addition $c_0 = a_0 + b_0$, während alle andern Felder ungenutzt bleiben. Basiert der Algorithmus aber auf Vektoroperationen, sind heute mit jedem gängigen Prozessor gleichzeitig vier oder mehr einfach genaue Rechenoperationen mit einem Befehl durchführbar. Zur Architektur-unabhängigen Nutzung solcher Funktionalität haben wir C++ Vektorklassen [5] entwickelt, die auch komplexe Ausdrücke erlauben, die zusätzlich mit Hilfe von maskierbaren Zuweisungen teure Sprungbefehle vermeiden.

## Vom Spielen zum Rechnen

Computergenerierte Grafik und Visualisierung benötigen immer höhere Parallelität, beispielsweise um die realistischen Licht- und Schattenverhältnisse für Computerspiele zu berechnen. Die dabei anfallenden aufwändigen Rechenoperationen lassen sich in der Regel parallel für verschiedene Segmente des hochauflösenden Bildschirms rechnen. Da diese Operationen nur für die Bildschirmausgabe relevant sind, sind Grafikkarten (Graphics Processing Unit, GPU) mit immer mehr parallel arbeitenden Rechenwerken zur Unterstützung des Prozessors ausgestattet. Das Rechenwerk ist aber der Kernbaustein jedes Prozessors und führt die eigentlichen arithmetischen und logischen Operationen aus, daher auch der Name „Arithmetic, Logic Unit" (ALU). Der Prozessor benötigt zusätzlich Instruktionsdekodierer, die Befehle in Steuerkommandos übersetzen, und Steuerwerke, die der ALU anzeigen, welche Operation sie gerade ausführen soll, und darüber hinaus die verschiedenen Unterfunktionen des Prozessors koordinieren. Allerdings wird in der GPU nur mit einfacher Genauigkeit gerechnet, doppelte Genauigkeit lässt sich durch Zusammenschalten mehrerer Rechenwerke erreichen.

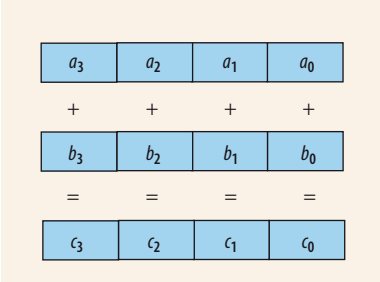



**Abb. 3** In einem 128-Bit-Register lässt sich mit einem Befehl auch eine Vektoraddition von vier 32-Bit-Zahlen in einfacher Genauigkeit ausführen.

Grafikkarten für PCs werden heute in großen Stückzahlen produziert. Daher sind leistungsfähige Grafikkarten schon für weniger als 500 Euro verfügbar. Solche GPUs können mehr als 500 Milliarden doppelt genaue und mehr als 2500 Milliarden einfach genaue Fließkomma-Rechenoperationen pro Sekunde (500 bzw. 2500 Gigaflops) durchführen. Im Vergleich dazu liefert der Prozessorkern eines PCs knapp 10 Gigaflops bei doppelter Genauigkeit. Der große Rechenleistungsunterschied zwischen einfacher und doppelter Genauigkeit legt nahe, so weit wie möglich einfach genau zu rechnen, was den zusätzlichen Vorteil des halben Speicherbedarfs bringt.

Die Architektur einer Grafikkarte verfügt über verschiedene Speicherbereiche und Multiprozessoren, die mehr als tausend Rechenwerke implementieren können (**Abb. 4**). Der Hauptspeicher der GPU erlaubt sehr hohe Zugriffsraten von etwa 150 Gigabyte pro Sekunde, während Prozessoren etwa 20 GB/s erreichen. Vor der Berechnung muss der PCI-Express Bus die Daten in die GPU kopieren, was mit Bandbreiten von etwa 6 GB/s möglich ist. Danach kann eine Berechnung starten. Die Ergebnisse müssen wieder in den Hauptspeicher des Rechners zurückkopiert werden. Diese Kopiervorgänge können bei geschickter Programmierung allerdings mit anderen Rechnungen überlappen, benötigen also nur wenig zusätzliche Zeit.

Die Rechenwerke einer GPU sind in Multiprozessoren zusammengefasst, die völlig unabhängig

voneinander operieren und über den GPU-Hauptspeicher (Dynamic Random Access Memory, DRAM) kommunizieren. Sie implementieren zusätzliche lokale Speicher und Cache-ähnliche schnellere Speicher. Innerhalb eines Multiprozessors führen die Rechenwerke immer den gleichen Befehl aus, da nur ein Instruktionsdekodierer vorhanden ist. Der Instruktionsstrom einer solchen ALU heißt Thread (engl. für Strang oder Faden), da derselbe Befehl mehrfach unabhängig auf unabhängigen Registern ausgeführt wird. Im Gegensatz zum Vektorbefehl kann jede ALU auf ihren eigenen Registern operieren, wie in **Abb. 4** angedeutet, und hat somit im Vergleich zu den Vektoroperationen wesentlich mehr Freiheitsgrade bei den Operanden. Diese Prozessorgruppe heißt daher „Single Instruction – Multiple Thread" (SIMT). SIMT bezeichnet also die parallele mehrfache Ausführung derselben arithmetischen Befehle auf verschiedenen ALUs mit unabhängigen Registern, während SIMD die parallele Ausführung von Befehlen einer ALU auf unterteilten Registern bedeutet (vgl. **Abb. 3**).

Deutlich wird dabei, wie wichtig inhärente Parallelität in der Anwendung ist, um entweder SIMD- oder SIMT-Architekturen zu verwenden. Wenn ein Problem gut vektorisierbar ist, dann ist es auch sehr gut auf SIMT übertragbar, aber nicht umgekehrt.

## Grafikprozessor als Rechenbeschleuniger

Die gewaltigen Rechenanforderungen bei den geschilderten Teilchenexperimenten legen den Einsatz von Grafikkarten (GPU) als Rechenbeschleuniger nahe, zumal ihre Anschaffungs- und Betriebskosten wesentlich günstiger sind als für herkömmliche Computersysteme. Mittlerweile ist es gelungen, auf Basis von GPUs Beschleunigungen von mehreren Größenordnungen zu erreichen [6]. Doch hierzu waren einige Hürden zu nehmen:

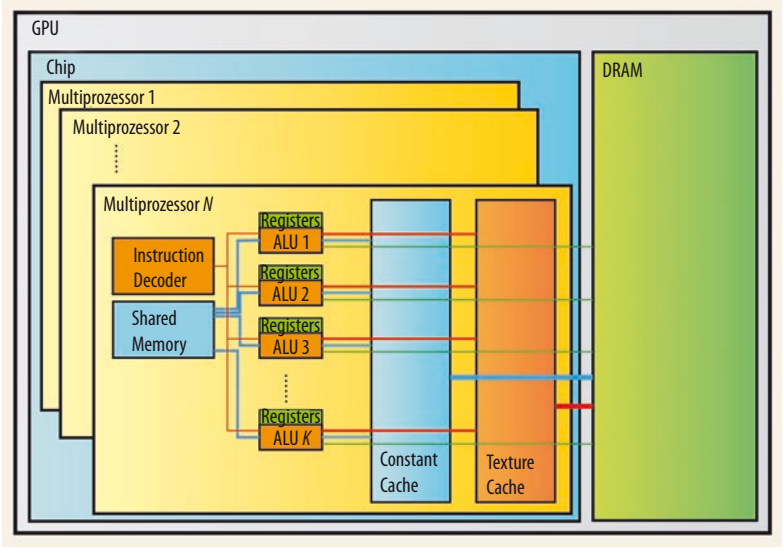



**Abb. 4** Vereinfacht dargestellt besteht die Architektur einer Graphikkarte (GPU) aus mehreren (*N*) unabhängigen „Multiprozessoren". Jeder Multiprozessor enthält einen Befehlsdekodierer (instruction decoder) und viele (*K*) Rechenwerke (ALUs), die alle denselben Befehl ausführen müssen, aber als Argumente eigene Register haben. Jeder Multiprozessor besitzt zusätzlich schnelle lokale Speicher, die direkt zugreifbar sind, kann aber auch auf den gemeinsamen Hauptspeicher der GPU (DRAM) zugreifen. Diese Zugriffe sind aber in Konkurrenz mit allen anderen Multiprozessoren.

Da die Teilchenspuren durch ein starkes solenoides Magnetfeld gekrümmt werden, um den Teilchenimpuls bestimmen zu können, muss an jeder Stelle des gesamten TPC-Detektorvolumens das präzise vermessene Magnetfeld bekannt sein. Daraus ergeben sich Tabellen der Magnetfeldwerte von rund 10 MB. Die durch die Teilchenspur vorgegebenen Zugriffsmuster auf diese Tabellen sind jedoch rein zufällig und erzeugen fast ausschließlich Cache Misses. Um diese zu vermeiden, reduziert die Interpolation durch Polynome die Magnetfeldtabelle drastisch auf wenige Koeffizienten. Die benötigten Werte werden also erneut berechnet, anstatt sie aus dem Speicher zu beziehen. Die sich dabei ergebenden Fehler wirken sich nicht auf die Genauigkeit der bestimmten Impulse aus. Obwohl nun mehr gerechnet werden muss, um den Magnetfeldvektor an einem bestimmten Raumpunkt zu erhalten, beschleunigt sich die Magnetfeldauslese um einen Faktor 39.

Im Zusammenhang mit der Spurrekonstruktion sinkt die Zugriffszeit auf die Magnetfelddaten sogar auf Null, da diese Interpolationen im Hintergrund stattfinden können. Das ist der höheren Rechengeschwindigkeit aufgrund der größeren Parallelität zu verdanken. Gleichzeitig ist es auch möglich, mehrere Spurrekonstruktionen gleichzeitig auf Mehrkernprozessoren laufen zu lassen, ohne dass sie sich gegenseitig durch die begrenzte Speicherbandbreite behindern. Diese Optimierung demonstriert als ein zentraler Punkt den Paradigmenwechsel der Programmierung auf modernen Multi- und Manycore-Architekturen. Speicherzugriffe sind nach Möglichkeit auch auf Kosten höheren Berechnungsaufwandes zu vermeiden.

Der konventionelle Kalman-Filter benötigt normalerweise doppelte Rechengenauigkeit, da sonst numerische Fehler zu groß ausfallen. GPUs erreichen aber ihre höchste Rechenleistung nur in einfacher Rechengenauigkeit. Verschiedene Ansätze für die Lösung dieses Problems sind denkbar, wie der Einsatz von doppelt genauen Variablen in kritischen Bereichen, etwa der Kovarianzmatrix. Bei genauer Studie des Kalman-Filters mit Spurrekonstruktionsdaten zeigt sich aber, dass die größten numerischen Fehler während der ersten Iterationsschritte auftreten. Hier sind die Fehler der zu berechnenden Spurparameter noch sehr hoch. Aber durch geschickte Maskierung besonders großer Fehlerausreißer war es möglich, diese numerischen Instabilitäten zu beseitigen.

Weitere mathematische Optimierungen ermöglichen hier eine Geschwindigkeitssteigerung um einen Faktor von 6,4. Diese weiteren Optimierungen sind bei den verschiedensten Anwendungen nützlich, sind aber auch beim Implementieren des Algorithmus zu berücksichtigen. Das setzt ein Mindestverständnis der Prozessor- bzw. Rechenwerkarchitektur voraus. Der Umfang dieses Artikels erlaubt nicht, hier auf Details einzugehen. Deshalb seien hier nur einige Stichpunkte

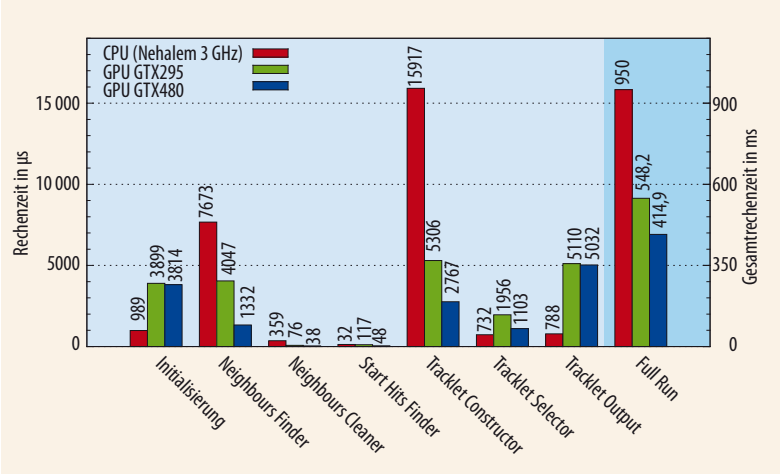



**Abb. 5** Für eine simulierte Kollision von Blei-Ionen lässt sich die Leistungsfähigkeit von CPU (mit vier Prozessoren, rot) und GPU (blau bzw. grün) anhand der Zeit (in µs) vergleichen, die für die verschiedenen Rechenschritte nötig sind. Diese sind in ihrer Reihenfolge im Programm von links nach rechts angeordnet. Besonders schnell sind die GPUs beim Finden der Nachbar-Hits und bei der Spurkonstruktion. Ganz rechts ist die Gesamtzeit (ms) aller Rechenschritte dargestellt.

exemplarisch genannt: Reduktion der Matrixoperationen zu direkten Operationen mit nicht trivialen Elementen, explizites „loop unrolling", Sprungvermeidung, Zusammenfassen von Funktionsaufrufen, Neuordnung der Berechnungen zur optimalen Ausnutzung der Prozessorpipeline.

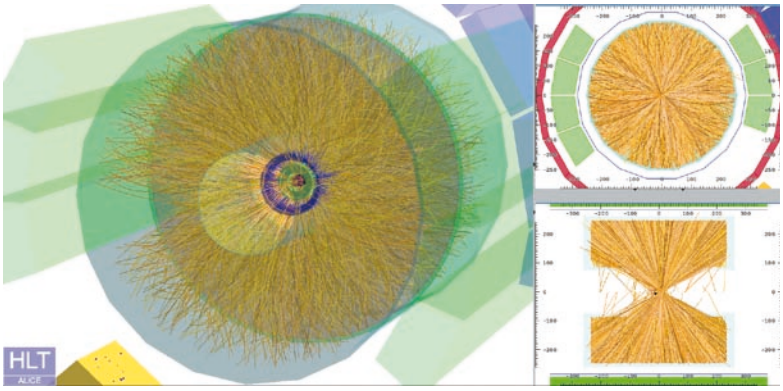
Ein anderer wichtiger Schritt, um die Rechenleistung zu optimieren, ist die Vektorisierung. Nachdem alle Spuren unabhängig sind, lassen sie sich entsprechend unabhängig berechnen. So werden dann aus den Ortskoordinaten Vektoren mit $k$ Ortskoordinaten, wobei $k$ die Anzahl der Elemente pro Vektor ist (derzeit vier bei Intel- oder AMD-Prozessoren). Hier kommen die besprochenen Vektorklassen zum Einsatz [5]. Diese Vektorisierung ergab eine weitere unabhängige Geschwindigkeitssteigerung um den Faktor 3,7. Die über alles gemittelte Geschwindigkeitssteigerung, die sich nur durch den optimierten Algorithmus für moderne Computer ergibt, beläuft sich auf den Faktor 40. Von besonderer Bedeutung ist die Tatsache, dass all diese Optimierungen, die ursprünglich für GPUs gedacht waren, gleichermaßen auf jedem marktüblichen Intel- oder AMD-Prozessor realisierbar sind.

Der nächste Schritt ist es, die Spurrekonstruktion auf einer GPU tatsächlich durchzuführen. Hierbei müssen die Daten und alle notwendigen Konfigurationsparameter zunächst in den Speicher der GPU geladen werden, um dann den Algorithmus auszuführen. Die zwei Endkappen der TPC (vgl. **Abb. 1**) bestehen aus jeweils 18 unabhängigen Sektoren, die sich auch unabhängig berechnen lassen, da keine relevanten Teilchenspuren von einem Sektor zu einem anderen wechseln können. Das erlaubt eine Pipelineverarbeitung der einzelnen Sektoren. Teile der Rekonstruktionssoftware laufen effizienter auf CPUs, wie später gezeigt wird, andere sehr viel besser auf GPUs. Deshalb können nach der Initialisierung sowohl die CPU als auch die GPU verschiedene TPC-Sektoren gleichzeitig berechnen, während zusätzlich relevante Daten von der GPU direkt in den Hauptspeicher übertragen werden.

Für die eigentliche Berechnung steht im Vergleich zum Prozessor wesentlich mehr Parallelität zur Verfügung, die es aber auch entsprechend zu nutzen gilt. Die Spurrekonstruktion besteht aus den Einzelschritten „Neighbours Finder", „Tracklet Constructor" und „Tracklet Selector", wobei der „Tracklet Constructor" den mit Abstand größten Anteil der Rechenzeit erfordert. Hier wird jeweils eine Spur pro GPU-Thread berechnet. Das führt aber dazu, dass alle anderen Threads eines Multiprozessors warten müssen, bis die längste Spur fertig berechnet ist. Das bedeutet große Ineffizienzen (Recheneffizienz < 20 %).

Wesentlich effizienter ist es, alle Spuren nur bis zu einer bestimmten Maximallänge zu berechnen und danach alle noch unfertigen Spursegmente mit einem hierfür speziell entwickelten Scheduler neu auf die Rechenwerke zu verteilen. Ein Vorfilter entfernt dabei besonders kurze Spursegmente aus den Listen. Für höchste Effizienz bearbeitet der Scheduler mehrere TPC-Segmente gleichzeitig, sodass immer genügend Spurdaten zur Verarbeitung stehen. Damit lässt sich eine Effizienz von 70 Prozent erreichen.

Die weiteren Rechenschritte „Neighbours Finder" und „Tracklet Selector" werden ebenfalls auf der GPU ausgeführt, um den Datenaustausch zwischen Host und GPU zu minimieren. So ist der „Tracklet Selector" auf der GPU im Vergleich zur CPU langsamer, aber er reduziert die Größe des Datensatzes erheblich, wodurch es trotzdem ökonomisch ist, ihn auf der GPU auszuführen, da weniger Daten zu übertragen sind.




ALICE/CERN

**Abb. 6** Die Visualisierung einer im High Level Trigger (HLT) online rekonstruierten Blei-Blei-Kollision (event display) bei 2,76 ATeV zeigt eine perspektivische Darstellung der TPC-Spuren (links). Man erkennt deutlich die verschiedenen gekrümmten Spuren. Rechts oben ist die Frontalansicht auf die Endkappe der TPC zu erkennen und rechts unten die entsprechende Seitenansicht. Man sieht hier sehr deutlich, dass die Spuren von einem Vertex entstammen.

**Abb. 7** Der neue Hochleistungsrechner „LOEWE-CSC" der Universität Frankfurt enthält insgesamt 772 GPUs und 20 928 Prozessorkerne. Auf der Liste der weltweit schnellsten Großrechner belegt er Platz 22 und Platz 8 auf der Liste der energieeffizientesten Rechner.

Die Initialisierung und Ausgabefunktionen dagegen enthalten viel zufällig verteilte Speicherzugriffe, die von großen, hoch entwickelten Caches profitieren können, weshalb sie sich besser von einer CPU ausführen lassen. Die GPU-Anpassung der Algorithmen optimiert das Speichermodell außerdem in Bezug auf höhere Lokalität und effiziente Speicherausnutzung. Das hatte auch einen positiven Effekt auf die entsprechende CPU-Version. Für simulierte Blei-Ereignisse wurde mit der optimierten CPU-Version verglichen (**Abb. 5**).

## Erste Ergebnisse und neue Perspektiven

Alle besprochenen Aspekte der Computerarchitektur und der schnellen Ereignisrekonstruktion wurden beim ALICE High Level Trigger berücksichtigt, der seit Ende 2009 im Dauereinsatz ist und natürlich auch die ersten Blei-Blei-Kollision in ALICE im November 2010 rekonstruierte (**Abb. 6**). Die verschiedenen Spuren sind klar zu erkennen. Aufgrund der noch niedrigen Kollisionsrate ließ sich hier allerdings nicht die volle Rechenleistung des High Level Triggers mit einer Ereignis-Verarbeitungsrate von mehr als 100 Hz ausnutzen. Basierend auf den rekonstruierten Ereignissen wird nun gerade begonnen, erste Algorithmen zu implementieren, die beispielsweise Ereignisse mit Teilchen selektieren, die einen großen Transversalimpuls aufweisen.

Andere Experimente haben diese erfolgreichen Ansätze nun auch übernommen. So ist es geplant, die ALICE-Ereignisrekonstruktion zukünftig bei STAR in Brookhaven einzusetzen [7]. Gleichzeitig kommen diese Prinzipien bei den geplanten Experimenten bei FAIR zum Einsatz [8]. Wenn auch die hier besprochenen Anwendungen ein spezielles Gebiet der Physik betreffen, sind doch die Prinzipien allgemein gültig. Kalman-Filter finden auch ein weites Anwendungsfeld in Wissenschaft und Industrie. Wir planen, die bei der Optimierung des Kalman-Filters gewonnene Erfahrung hier einzubringen.

So wurden erste neue Versionen der Frankfurter ultrarelativistischen Quantenmolekulardynamik-Software und eine erste Version eines Gitter-QCD-Programms entwickelt, das auf mehreren GPUs verteilt lauffähig ist. Hier wurden dieselben grundlegenden Prinzipien der Erzeugung hoher Lokalität und Unabhängigkeit der Daten bei gleichzeitig möglichst hoher inhärenter Parallelität angewendet.

Der Trend der Computerindustrie hin zu mehr Parallelität wird sich weiter fortsetzen. Insofern wird es immer wichtiger, bei der Softwareentwicklung diese Paradigmenveränderungen zu berücksichtigen. Wenn man die Amortisationszeit der gesparten Kosten für die benötigte Rechenzeit berücksichtigt, lohnt sich auch schnell ein größerer Entwicklungsaufwand. Um diesen Prozess zu beschleunigen und die Anwender auf dem Weg zu moderner Programmierung zu unterstützen, wurden in Frankfurt so genannte SimLabs eingerichtet, in denen Wissenschaftler aus den Anwendungen und den Computerwissenschaften möglichst interdisziplinär eng an der Weiterentwicklung der Algorithmen arbeiten. Als zukünftige Arbeitsplattform in diesem Kontext bewilligte die DFG den fünf Millionen Euro teuren neuen Hessischen Hochleistungsrechner „LOEWE-CSC" der Goethe-Universität Frankfurt (**Abb. 7**). Damit steht nun ein System zur Verfügung, das es erlaubt, auch sehr komplexe Probleme auf großen Skalen zu rechnen.

*

Danken möchte ich Sergey Gorbunov, Mathias Kretz und David Rohr, die zu den hier genannten Arbeiten wesentlich beigetragen haben.

### Literatur

[1] *ALICE Collaboration*, The ALICE Experiment at the CERN LHC, JINST 3:S08002 (2008)
[2] *V. Lindenstruth* und *I. Kisel*, Nucl. Instrum. Meth. A**535**, 48 (2004)
[3] *Konrad Zuse*, Rechnender Raum, Vieweg, Braunschweig (1969)
[4] *R. E. Kalman*, Trans. ASME J. Basic. Eng., D **82**, 35 (1960)
[5] *M. Kretz*, Efficient Use of Multi- and Many-Core Systems with Vectorization and Multithreading, http://compeng.uni-frankfurt.de/index.php?id=vc
[6] *S. Gorbunov, U. Kebschull, I. Kisel, V. Lindenstruth* und *W. F. J. Müller*, Computer Physics Communications 178, 374 (2008)
[7] *Star Collaboration*, STAR Detector Overview, Nucl. Instrum. Meth. A **499**, 624 (2003)
[8] FAIR – Baseline Technical Report (2006), www.gsi.de/documents/DOC-2006-Sep-12-1.pdf

### DER AUTOR

**Volker Lindenstruth** (FV Hadronen und Kerne) studierte Physik in Darmstadt und promovierte 1993 an der Universität Frankfurt im Bereich der Schwerionenphysik. Anschließend war er bis 1995 als Feodor von Lynen-Fellow für Informatik in den USA. Am Lawrence Berkeley National Laboratory in Kalifornien forschte er als Postdoc in der Abteilung „Nuclear Science". 1998 nahm er einen Ruf nach Heidelberg an und vertrat dort das Fach technische Informatik. Seit dieser Zeit leitet er auch den ALICE High Level Trigger. Seit 2009 hält er die LOEWE HIC for FAIR-Professur an der Goethe-Universität Frankfurt. Der Schwerpunkt seiner Forschung liegt auf der Entwicklung Energie-effizienter Computerarchitekturen und -algorithmen.