Exascale-Computing. Algorithmen.

Die Einleitung

Die Entwicklung von Computern befindet sich im ständigen und schnellen Wachstum. Für wissenschaftliches Rechnen jedoch sind die einfachen PCs nicht mehr geeignet, denn zum Hochleistungsrechnen braucht man einen Supercomputer. In Wikipedia werden er als *„die schnellsten Rechner ihrer Zeit bezeichnet. Dabei ist es unerheblich zu wissen, auf welche Bauweise der Rechner beruht, solange es sich um einen universal einsetzbaren Computer handelt"*[1] bezeichnet. Aber was ist Exascale Computing? Und wie unterscheidet es sich von anderen Supercomputern?

Die Antwort liegt in ihrem Name: „Exascale". So ist „Exa" ein Präfix für 10_{18} und genau so viele Gleitkommazahl-Operationen müssen Exascale-Computer tun können. Zum Beispiel, der Prozessor Core i7, 3,2 GHz mit 4 Kerne hat 51,2 GFLOPs (FLOP – eng. Floating Point Operations Per Second, Gleitkommazahl-Operation) in Hochleistung [2]. Alle Supercomputers und natürlich auch Exascale sind ziemlich groß und verbrauchen relativ viel Strom. Wir müssten die Kosten minimalisieren. Es ist aber sehr schwer und kompliziert, Preise zu reduzieren. Trotzdem können wir sehr effiziente Algorithmen nutzen, die weniger Operationen brauchen. Darüber werden wir hier sprechen.

Die Tasks

Zuerst müssen wir klären, wofür es effizient ist einen Supercomputer zu nutzen und wofür nicht. Supercomputer sind für Hochleistungsrechnen gedacht, das bedeutet, dass für solche Computer einfache Aufgaben wie z.B. Office-Arbeit und Buchhaltung nicht angemessen sind. Viel eher sind sie zur Lösung schwerer Aufgaben wie z.B. dem Lösen von komplexen Gleichungssystemen und dem N-Body Problem, geeignet.

Gleichungssystemen Lösung

Zur Lösung von Gleichungssystemen nutzt man am besten das Gaußsche Eliminationsverfahren. Es ist sehr einfach und sieht so aus [3]:
Gesucht sei die Lösung für das lineare Gleichungssystem:

$$6 = -12z - 10x$$
$$-5x = -17 - 4w + 8z + 2y$$
$$23 = 4z + 5x$$
$$-10x - 16z = 2 - 4w$$

Zuerst zieht man alle Variablen nach links und alle Konstanten nach rechts.

$$
\begin{aligned}
10x+12z&=-6\\
4w-5x-2y-8z&=-17\\
-5x-4z&=-23\\
4w-10x-16z&=2
\end{aligned}
$$

Man vereinfacht sich die Schreibarbeit, indem man nur die Zahlen erfasst, also die Koeffizienten aus der linken Seite und die absoluten Zahlen aus der rechten Seite der Gleichungen. Dabei müssen die Vorzeichen beachtet und übernommen werden. Für alle fehlenden Variablen wird eine Null geschrieben.

$$
\begin{matrix}
0 & 10 & 0 & 12 & -6\\
4 & -5 & -2 & -8 & -17\\
0 & -5 & 0 & -4 & -23\\
4 & -10 & 0 & -16 & 2
\end{matrix}
$$

Dieses System nennt man eine Matrix. Weiter man muss durch geeignete Umformungen in die Form versuchen:

$$
\begin{matrix}
1 & 0 & 0 & 0 & a\\
0 & 1 & 0 & 0 & b\\
0 & 0 & 1 & 0 & c\\
0 & 0 & 0 & 1 & d
\end{matrix}
$$

In unserem Beispiel bekommt man:

$$
\begin{matrix}
1 & 0 & 0 & 0 & -14\\
0 & 1 & 0 & 0 & 15\\
0 & 0 & 1 & 0 & -5\\
0 & 0 & 0 & 1 & -13
\end{matrix}
$$

was bedeutet

$$w=-14, x=15, y=-5, z=-13$$

Aber hier treffen wir eine Störung. Für eine Matrix NxN braucht man ca. $O(N^3)$ Operationen. Das bedeutet, dass für eine Matrix mit $N=10^6$ es 10^{18} Operationen ist. Ein PC mit 1 TFLOPs kann dass in 11,5 Tage berechnen. Es ist zu viel für uns, aber es gibt eine Alternative – Multigrid.

Multigrid als eine Lösung für Gleichungssystemen

Mehrgitterverfahren (eng. Multigrid) ist ein effizienter Algorithmus zur näherungsweisen Lösung von Gleichungssystemen. Er passt nicht nur für klassische Gleichungssysteme, sondern auch für elliptische Gleichungen. Multigrid nutzt eine Folge abnehmbarer Gitter und den Korrektur-Operator. Wir können unbekannte Fehler zu einer gegebenen Näherung auf einem feinen Gitter approximieren. Es ist

günstig, weil auf dem groberen Gitter weniger Unbekannte gegeben sind. Die Kombination von Grobgitterkorrektur und Glatter gibt eine schnelle Konvergenzrate.

Multigrid

Die Hauptidee einer Approximation von unerkannten Fehlern ist die, der unerkannten Fehler auf einem freien Gitter und unerkannten Fehler auf einem gröberen Gitter. Auf den gröberen Gittern sind weniger unbekannte Variablen, daher ist es im Rahmen der Komplexität einfacher und günstiger. Durch rekursive Prozesse von gröber werdenden Gittern bekommen wir eine Multigrid-Struktur.

Die Benutzung der groben Gitter macht Berechnung schneller über Diskretisierung. Die Grobgitterkorrektur nivelliert Fehlverhalten.

Zuerst sei auf jedem Gitter (Bild 1) ein lineares Gleichungssystem $Au=f$, wo A eine reguläre quadratische Matrix, $A \in \mathbb{R}^{NxN}$, mit den Elementen a_{ij} ist. Wir korrelieren Indexen mit Knoten. D.h. u_i ist u in der Knote i. Menge der Knoten vom Gitter geben $\Omega = \{1,2,\ldots,n\}$ vor. Wichtig ist, dass Fehler e vom Gitter gelöscht werden kann, mit Hilfe der Lösung auf groben Gitter.

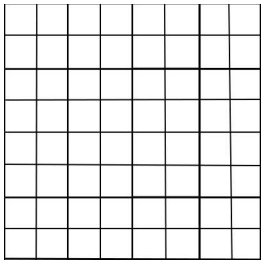

Bild 1. Der Gitter.

Auf der erste Etappe machen wir Gittern grober (Bild 2) und mit jedem Zyklus berechnen wir auch Grobgitterkorrektur, Prolongationen und Nachglättungen. Mit rekursivem Aufruf wiederholen wir schließlich reichlich grobe Gitter haben.

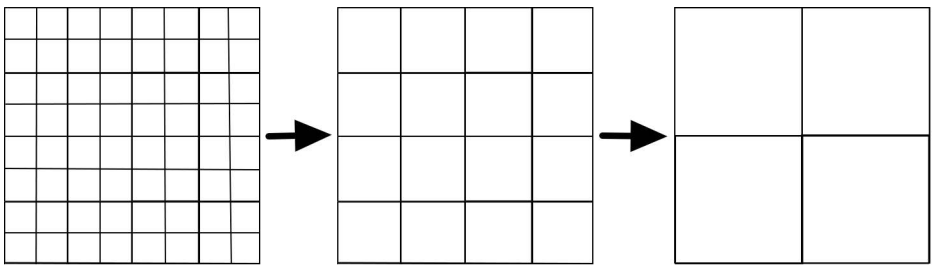

Bild 2. Prozess der Verrohung.

Danach berechnen wir unsere Gleichungssysteme für die groben Gitter. Die Lösung des letzten Gitteres ist der letzte rekursiver Aufruf. Weiter gehen wir durch die Rekursion zu den vorherigen Gittern zurück. In jedem Schritt ziehen wir die berechneten Werte von den folgenden Gittern ab und setzen Korrekturen ein und berechnen das aktuelle Gitter. Und so kehren wir zum ersten Gitter zurück (Bild 3).

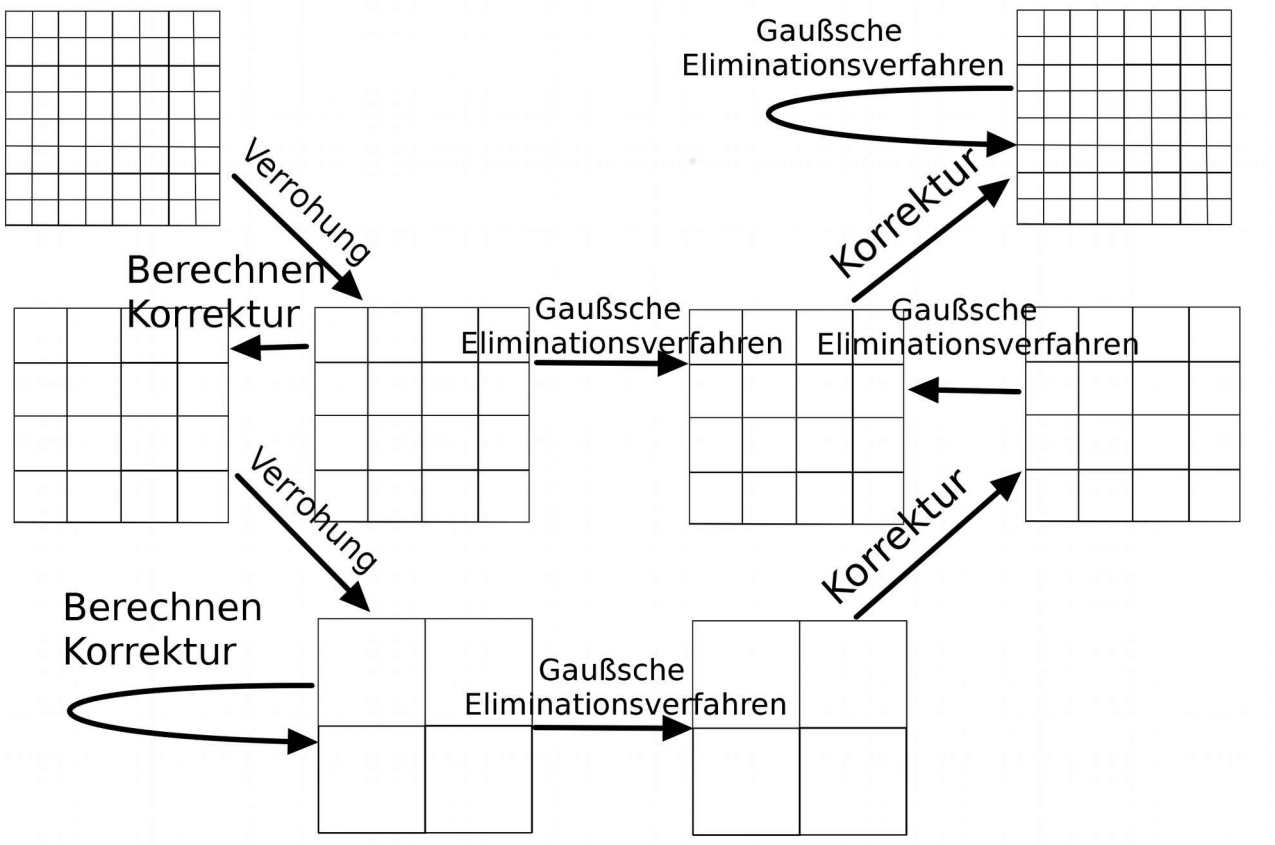

Bild 3. Multigrid.

Warum ist Multigrid besser als Gauß-Methode?

Nach dem Lesen der Algorithmus-Beschreibung könnte noch unklar bleiben, warum die Multigrid-Methode besser ist als die Gauß-Methode, dafür gibt es 2 Hauptgründe:

Multigrid viel schneller. Je nach Operator und Variant des Algorithmus, kann die Laufgeschwindigkeit von O(NlogN) bis O(N) variieren. Dies hat auch William L. Briggs sehr gut in seiner Vorlesung erklärt [4]. Zum anderen ist ein Multigrid sehr flexibel. Wir wählen unsere Operatoren und wählen damit auch gleichzeitig die Konvergenz und die Komplexität.

Natürlich hat gleichzeitig Multigrid die Nachteile und am wichtigsten für uns und Exascale ist die Größe der Speicher. Je mehr Iterationen, desto mehr Speicher brauchen wir. Seit 60er wurde der Algorithmus stark entwickelt. Es gibt heute viele Variationen: klassisches Multigrid, paralleles Multigrid und viele andere. Er kann immer weiter entwickelt werden, um den Bedürfnissen der Mathematikers gerecht zu werden.

Mehrkörper-Problem (Dreikörperproblem) und seine Lösung

„Das Dreikörperproblem der Himmelsmechanik besteht darin, den Bahnverlauf dreier Körper unter dem Einfluss ihrer gegenseitigen Anziehung (Gravitation) vorherzusagen. Um quantitative Resultate zu erlangen, muss es im allgemeinen Fall bislang nummerisch gelöst werden.“ [Zit5] Eine Lösung für Mehrkörper-Probleme spielt eine sehr wichtige Rolle für die moderne Wissenschaft. Am meisten – für die Physik und die Mechanik. Der Nobelpreis für Chemie im Jahr 2013[8] bekräftigt diese Aussage. Seit 17. Jahrhundert ist es eines der schwierigsten mathematischen Probleme. Für einen Körper ist die Lösung das Beharrungsgesetz. Für zwei Körper ist das Problem durch die Keplerschen Gesetze lösbar und die Lösung stellt ein barazentrischer systematischer Orbit dar. Aber für $N \geq 3$ gibt es keine algebraische Lösung und die Funktion ist nicht lösbar [5]. Wahrscheinlich hat Karl Frithiof Sundman Anfang des 20. Jahrhundert eine analytische Lösung für drei Körper gezeigt, aber nur für ein Fall, und zwar, wenn der Gesamtdrehimpuls des Systems nicht verschwindet, was keine universale Lösung ist. Aber in den 80er Jahren haben Leslie Greengard und Vladimir Rokhlin „Fast Multipole Method“ erfunden, welche die Mehrkörperprobleme löst.

Fast Multipole Method

Ursprünglich wurde Fast Multipole Method (FMM) entwickelt, um ein Gravitationspotential oder ein elektrostatisches Potential für ein dreikörperiges Systeme zu berechnen. Aber es wurde sehr schnell klar, dass die Methode mehrere Anwendungen hat. Mithilfe Forscher von USA und Japan wurde FMM bei Aufgaben für Wärmedurchlässigkeit, Akustiker, Elektrodynamik und vielen andere eingesetzt. Außerdem wurde FMM für Fourierabbildungen, Interpolation, Gaussabbildungen usw abstrahiert.

FMM ist eine Methode, um die Summe aller Interaktionen zwischen Objekten im Raum zu berechnen.
Sei ein Raum mit Objekten (Bild 4).
x_i und y_i – Quellen und Rezipienten.
q_i – die Intensität der Quellen
K – Funktion, der Kern.
Und so unsere Summe muss so aussehen:

$$\Phi_j = \Phi(y_j) = \sum_{i=1}^{N} q_i K(y_j, x_i), j = 1..M \quad (1)$$

Das Berechnen der Funktion ist ein Produkt von einer Matrix mit Elementen K_{ij} auf Vektor q_i, was allgemein O(MN) Operationen braucht. Wir können die Anzahl der Operationen durch nächste Bewegungen abkürzen.

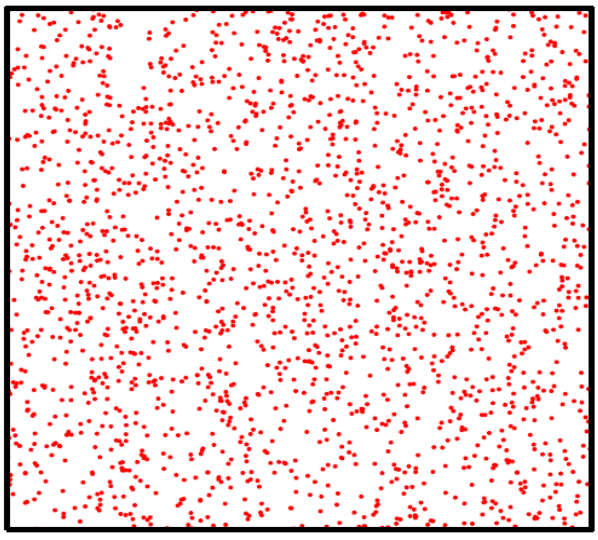

Bild 4. Das System[7].

1 Faktorisierung

Bei allgemeinem Beispiel die Formel (1) sieht so aus:

$$\Phi_j=\sum_{i=1}^{N} q_i(y_j-x_i)^2, j=1..M \quad (2)$$

Aber

$$(y_j-x_i)^2=(y_j^2-2x_i y_j+x_i^2)$$

Es ist günstiger eine Variable wie eine Konstante Berechnen. Stellen so x_i dar:

$$c_0=\sum_{i=1}^{N} q_i, c_1=\sum_{i=1}^{N} q_i x_i, c_2=\sum_{i=1}^{N} q_i x^2{}_i \quad (3)$$

Daher Formel 1 sieht so aus:

$$\Phi_j=c_0 y^2{}_j-2y_j c_1+c_2 \quad .$$

Die Koeffizienten C können für O(N) berechnet werden. Allgemein sieht das so aus: Lass K faktorisiert wie

$$K(y,x)=\sum_{n=1}^{\infty} C_n(x)F_n(y)=\sum_{n=1}^{P} C_n(x)F_n(y)+\varepsilon_p \quad , (4)$$

wo F_n – Basisfunktionen, C_n – Koeffizienten der Zerlegung. Wir nehmen an, dass Reihen konvergieren und bis zu den ersten P-Elementen abkürzt werden können. So bekommen wir einen Fehler ε_p. Stellen (4) in (1) und sehen, dass 3 weiter funktioniert und wir alle Φ_j in O(PM+PN) berechnen können! Wenn P sehr klein ist, ist es O(M+N).

2 Fehlerkontrolle

FMM hat eine Messabweichung. Je nach der Fehlkontrolle kann P eine Konstante sein oder mit N~M wie *logN* wechseln. Ebenfalls gibt es eine Möglichkeit, zwischen Exaktheit und Schnelligkeit wählen.

3 Multipole und lokale Zerlegungen

Die Summen (4) konvergieren nicht gleichmäßig und die Methode muss erweitert werden. Ersatzweise einen Bereich und eine Zerlegung behandeln wir viele Bereiche. Für jeden Bereich mit dem Zentrum in x_x bauen wir 2 Zerlegungen (für 2D):

$$K(y-x)=\sum_{n=1}^{P} C_n(x-x_x)S_n(y-x_x)+\varepsilon_p,\ |y-x_x|>\alpha|x-x_x|, \alpha>1 \quad (5)$$

$$K(y-x)=\sum_{n=1}^{P} D_n(x-x_x)R_n(y-x_x)+\varepsilon_p,\ |y-x_x|<\beta|x-x_x|, \beta<1 \quad (6)$$

Hier ist es die Zerlegung (5) - multipole (weiter – M-Zerlegung). Sie ist wahr in unendlichem weitem vom Zentrum der Zerlegung Bereich. Die Zerlegung (6) ist

wahr in einem endlichem Bereich von Zentrum und heißt lokale (weiter – L-Zerlegung). Basis S und R wählen wir wegen Konvergenz (was ist besser). Für 3D ist es typisch sphärische Funktion, für 2D – Kreisfunktion.

4 Raumunterteilung

Wir müssen auch unserer Raum unterteilen. Für einen 2D Raum stellen wir alles in einem Quadrat (3D – Würfel), dass wie weiter in viele kleinere Quadrate unterteilen.

Quadrate mit Quellen heißen Quellen, Quadrate mit Rezipienten – Rezipienten. Einige Quadrate können gleichzeitig mit Quellen und Rezipienten oder leer sein. Leere Quadrate können wir von Behandlung entfernen, was uns viel Zeit spart.

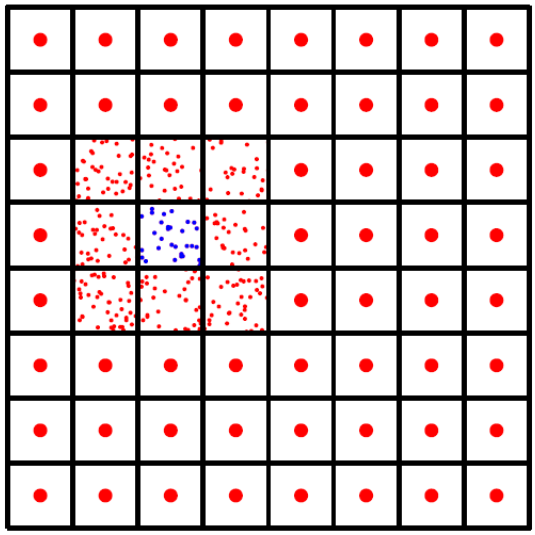

Bild 5. Raumunterteilung[7]

Für jede Quelle bauen wir eine M-Zerlegung. Die Zerlegung erklärt die Einfluss von allen Quellen in Quadrat auf alle Rezipienten außerdem Bereich des Quadrats (Bild 5). So forcieren wir unsere Addition. Aber wir können nicht O(N) bekommen, Maximum ist $O(N^{3/2})$. Translationen können uns helfen, um bis $O(N^{3/4})$ verbessern.

5 Translationen

Wenn wir eine M- oder eine L-Zerlegung in Raum Ω mit dem Zentrum x_x haben und Zerlegungen für Unterraum Ω‘ mit dem Zentrum $x_{‘x}$ suchen, können wir einfach Operator der Translation $T(x_{‘x} - x_x)$ zum Koeffizienten der Zerlegungen benutzen, um Koeffizienten für suchende Zerlegungen zu bekommen.

Operator der Translationen ist linear und kann für P Elementen mit P variablen als einer Matrix PxP

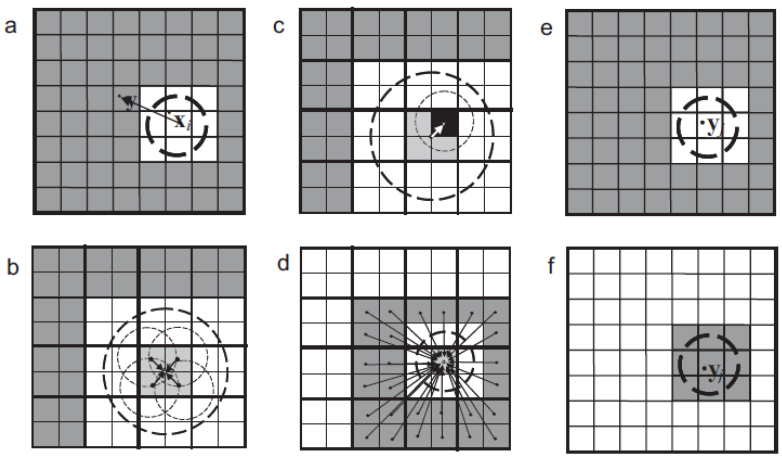

Bild 6 Etappe des Algorithmus [17]

vorgestellt werden. Eine Multiplikation einer Matrix auf einen Vektor braucht $O(P^2)$ Operationen. Aber wegen der Struktur von der Matrix ist es möglich in O(P) berechnen. In hierarchischer FMM benutzen wir drei Arte der Translationen: M2M, M2L und L2L, wo der erster Buchstabe Startzurlegung und die zweite – die letzte Zerlegung bedeutet (Bild 6: a) M-Zerlegung, b) M2M-Translations, a) L2L-Translations, d) M2L- Translations, e) L-Zerlegung, f) Summieren in Näherung).

6 Hierarchie

Einfache Unterteilung ist nicht genug, um O(N) zu bekommen. Dafür teilen wir mithilfe Baum-Strukture. Erster Quadrat (Level 0) teilen wir in 4 gleiche kleine Quadraten (Level 1). Das wiederholen wir für jeden Quadrat bis Niveau L_{max} (wird aus Überlegungen von Optimalisation oder Qualität ausgewählt). Für jedes Niveau gibt es Quellen, Rezipienten, Quadraten-Eltern und Quadraten-Kinder.

M-Zerlegungen für $L<L_{max}$ finden wir wie eine Summe von M2M Translationen von L+1. L-Zerlegungen berechnen wir auf folgende Weise:

- Jedes Quadrat bekommt Information über L2L Translationen L-Zerlegungen für Eltern-Quadrat.
- Das summieren wir mit der Information über Quellen, die in Bereich vom Eltern-Quadrat aber nicht im Bereich vom Quadrat sind. Berechnen das durch die Summe aller M2L Translationen.

Somit sieht der Algorithmus aus, wie ein Anstieg durch Hierarchie der Quellen von L_{max} bis L_0.

7 Indexbindung

Die Anforderung zur Schnelligkeit setzt schnelle und bequeme Indexbindung voraus. Wir nutzen die Morton‘s-Indexbindung. Sie läuft von Aufgabe Peano [9] aus. Das können wir in O(N) Operationen tun.
Die Anforderung zur Schnelligkeit setzt schnelle und bequeme Indexbindung voraus. Wir nutzen die Morton‘s-Indexbindung. Sie läuft von Aufgabe Peano [9] aus. Das können wir in O(N) Operationen tun.

Benutzung der FMM

Das waren die Hauptideen der FMM. Aber natürlich ist sie nicht perfekt. Ja wir können für viele verschiedene Lerne realisieren, wie z.B. RBF-Interpolation [10] (Kern K hangt von der Distanz zwischen Quellen und Rezipienten r=|y-x| ab. Das

benutzt man in Aufbau der Bilder [Bild 7]), Helmholtz-Gleichung [11, 13] usw. Auch wir können Das System als evolutionierende Gänze der Partikeln vorstellen (es ist sehr bequem, um Navier-Stokes Gleichungen [14, 15] zu berechnen) oder FMM für Randelementmethode [16] verwenden (Wir diskretisieren Fläche der Objekten mit N-Elementen, normalerweise wie ein Netz, und berechnen Integralgleichungen, was Integralgleichungen lineare macht. Die Randelementmethode ist ein sehr guter Beispiel, weil Randelementmethode sehr viele Elementen für Netze und als Folge viele Iterationen und Speicher braucht, was für Exascale perfekt passt.

Aber gleichzeitig ist FMM sehr kompliziert zu realisieren, was die Benutzung teuer und schwierig zu realisieren macht. Allgemein ist die FMM ein moderner aufstrebender Algorithmus, der viele Perspektiven hat und weiter entwickeln wird.

Bild 7 Benutzung FMM in CG [12]

Fazit

Die Algorithmen für Exascale sind ein sehr aktuelles Thema. Auf den ersten Blick kann es so scheinen, dass sie nicht so wichtig seien wie die technischen Aspekte. Aber es wurde deutlich gezeigt, dass es zumindest gleich wichtig ist. Die FMM und Multigrid bleiben nicht auf einem Platz stehen, sondern werden ständig weiterentwickelt. Gerade jetzt können wir nicht nur die Anwendungsmethoden lernen, sondern auch selbst zur Weiterentwicklung beitragen.

Quellen

[1] https://de.wikipedia.org/wiki/Supercomputer
[2] https://www.ibm.com/us-en/
[3] http://www.arndt-bruenner.de/mathe/9/lgsbsp2.htm
[4] “A Multigrid Tutorial“ William L. Briggs, SIAM, Philadelphia, 2000
[5] https://de.wikipedia.org/wiki/Dreikörperproblem
[7] "The Fast Multipole Method", Gunnar Martinsson, Dartmouth College, Dartmouth, 2014
[8] „US-Forscher Warshel, Karplus und Levitt erhalten Chemie-Nobelpreis 2013“ http://www.spiegel.de/wissenschaft/technik/nobelpreis-chemie-2013-geht-an-karplus-levitt-und-warshel-a-926912.html
[9] https://de.wikiversity.org/wiki/Aufgabenblatt_zu_Peano-Axiomen_1
[10] https://en.wikipedia.org/wiki/Radial_basis_function
[11] https://de.wikipedia.org/wiki/Helmholtz-Gleichung
[12] „Fast Multipole Method“, A.Gumerov, University of Maryland, Maryland 2013
[13] „A wideband fast multipole method for the Helmholtz equation in three dimensions“, Cheng H., Crutchfield W.Y., Gimbutas Z., Greengard L., Ethridge J. F., Huang J., Rokhlin V., Yarvin N., Zhao J., J. Comput. Phys. 2006
[14] https://en.wikipedia.org/wiki/Navier%E2%80%93Stokes_equations
[15] https://de.wikipedia.org/wiki/Navier-Stokes-Gleichungen
[16] „Principles of Boundary Element Methods“, Martin Costabel, Technische Hochschule Darmstadt
[17] "Multigrid Methods" Ulrich Rüde, Lehrstuhl für Systemsimlation Universität Erlangen-Nürnberg, Nürnberg, 2009