Partikelmethoden I + II

Stefan Lang

Interdisziplinäres Zentrum für Wissenschaftliches Rechnen
Universität Heidelberg
INF 368, Raum 532
D-69120 Heidelberg
phone: 06221/54-8264
email: Stefan.Lang@iwr.uni-heidelberg.de

WS 13/14

Themen

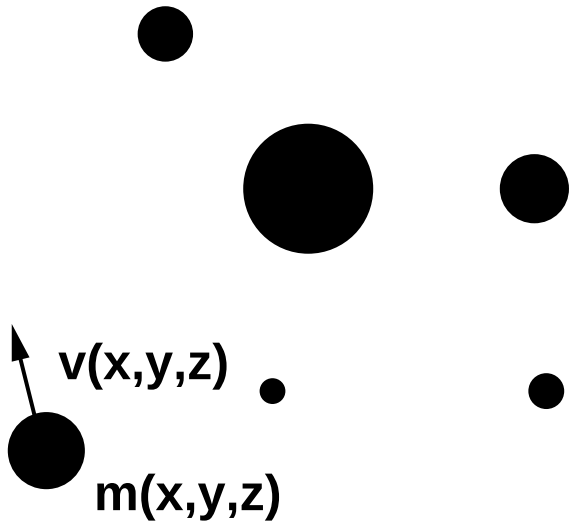
Partikelmethoden

- Problemstellung
- Standardverfahren
- Parallelisierung
- Verbesserung der Verfahren

Partikelmethoden

Aufgabenstellung:

- Mit Partikelmethoden simuliert man die Bewegung von N Teilchen (oder Körpern) welche sich unter dem Einfluss eines Kraftfeldes bewegen.
- Das Kraftfeld selbst hängt wieder von der Position der Teilchen ab.
- Die Teilchen sind durch Punktmassen $m(x, y, z)$ charakterisiert und bewegen sich mit Geschwindigkeit $v(x, y, z)$ im System.

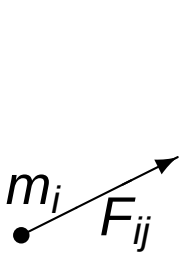


Das N-Körper-Problem I

N-Körper-Problem:

- Gegeben seien N punktförmige Massen $m_1, \ldots, m_N$ an den Positionen $x_1, \ldots, x_N \in \mathcal{R}^3$.
- Die von Körper j auf Körper i ausgeübte Gravitationskraft ist gegeben durch das Newton'sche Gravitationsgesetz

$$F_{ij} = \underbrace{\frac{\gamma m_i m_j}{\|x_j - x_i\|^2}}_{\text{Stärke}} \cdot \underbrace{\frac{x_j - x_i}{\|x_j - x_i\|}}_{\substack{\text{Einheitsvektor von } x_i \\ \text{in Richtung } x_j}} \tag{1}$$



- Die Gravitationskraft lässt sich auch als Potentialgradient schreiben:

$$F_{ij} = m_i \frac{\gamma m_j (x_j - x_i)}{\|x_j - x_i\|^3} = m_i \nabla \left(\frac{\gamma m_j}{\|x_j - x_i\|} \right) = m_i \nabla \phi_j(x_i). \tag{2}$$

$\phi_y(x) = \frac{\gamma m}{\|y - x\|}$ heißt Gravitationspotential der Masse m an der Position $y \in \mathcal{R}^3$.

Das N-Körper-Problem II

- Die Bewegungsgleichungen der betrachteten N Körper erhält man aus dem Gesetz Kraft=Masse$\times$Beschleunigung:

$$\text{für } i = 1, \ldots, N$$

$$m_i \frac{dv_i}{dt} = m_i \nabla \left(\sum_{j \neq i} \frac{\gamma m_j}{\|x_j - x_i\|} \right) \tag{3}$$

$$\frac{dx_i}{dt} = v_i \tag{4}$$

  Dabei ist $v_i(t) \colon \mathcal{R} \to \mathcal{R}^3$ die Geschwindigkeit des Körpers i und $x_i(t) \colon \mathcal{R} \to \mathcal{R}^3$ die Position in Abhängigkeit von der Zeit.

- Wir erhalten also ein System gewöhnlicher Differentialgleichungen der Dimension $6N$ (drei Raumdimensionen) für dessen Lösung noch Anfangsbedingungen für Position und Geschwindigkeit erforderlich sind:

$$x_i(0) = x_i^0, \quad v_i(0) = v_i^0, \quad \text{für } i = 1, \ldots, N \tag{5}$$

Numerische Berechnung

- Die Integration der Bewegungsgleichungen (3) und (4) erfolgt numerisch, da nur für $N = 2$ geschlossene Lösungen möglich sind (Kegelschnitte, Kepler'sche Gesetze).
- Das einfachste Verfahren ist das explizite Euler-Verfahren:

$$\text{Diskretisierung in der Zeit:} \quad \begin{aligned} t^k &= k \cdot \Delta t, \\ v_i^k &= v_i(t^k), \\ x_i^k &= x_i(t^k). \end{aligned}$$

$$\left.\begin{aligned} v_i^{k+1} &= v_i^k + \Delta t \cdot \nabla\left(\sum_{j \neq i} \frac{\gamma\, m_j}{\| \underbrace{x_j^k - x_i^k}_{\text{„explizit“}} \|} \right) \\ x_i^{k+1} &= x_i^k + \Delta t \cdot v_i^k \end{aligned}\right\} \text{ für } i = 1, \ldots, N \qquad (6)$$

Problematik der Kraftauswertung

- Sicher gibt es bessere Verfahren als das explizite Eulerverfahren, das nur von der Konvergenzordnung $O(\Delta t)$ ist, für die Parallelisierung spielt dies jedoch keine große Rolle, da die Struktur anderer Verfahren ähnlich ist.
- Das Problem der Kraftauswertung:
  In der Kraftberechnung hängt die Kraft auf einen Körper i von der Position *aller* anderen Körper $j \neq i$ ab. Der Aufwand für eine Kraftauswertung (die mindestens einmal pro Zeitschritt notwendig ist) steigt also wie $O(N^2)$ an. In den Anwendungen kann $N = 10^6, \ldots, 10^9$ sein, was einen enormen Rechenaufwand bedeutet.
- Ein Schwerpunkt dieses Kapitels ist daher die Vorstellung verbesserter sequentieller Algorithmen zur schnellen Kraftauswertung. Diese berechnen die Kräfte näherungsweise mit dem Aufwand $O(N \log N)$ (Es gibt auch Verfahren mit $O(N)$ Komplexität, die wir aus Zeitgründen weglassen).

Parallelisierung des Standardverfahrens

- Der $O(N^2)$-Algorithmus ist recht einfach zu parallelisieren. Jeder der P Prozessoren erhält $\frac{N}{P}$ Körper. Um die Kräfte für alle seine Körper zu berechnen, benötigt ein Prozessor alle anderen Körper. Dazu werden die Daten im Ring zyklisch einmal herumgeschoben.
- Analyse:

$$
\begin{aligned}
T_S(N) &= c_1 N^2 \\
T_P(N,P) &= c_1 P \Big(\underbrace{\frac{N}{P} \cdot \frac{N}{P}}_{\substack{\text{Block } p \\ \text{mit } q}} \Big) + \underbrace{c_2 P \frac{N}{P}}_{\text{Kommunikation}} = \\
&= c_1 \frac{N^2}{P} + c_2 N \\
E(P,N) &= \frac{c_1 N^2}{\left(c_1 \frac{N^2}{P} + c_2 N \right) P} = \frac{1}{1 + \frac{c_2}{c_1} \cdot \frac{P}{N}}
\end{aligned}
$$

konstante Effizienz für $N = O(P)$.

- Damit ist die Isoeffizienzfunktion wegen $W = c_1 N^2$

$$W(P) = O(P^2)$$

(natürlich in Bezug auf den suboptimalen Standardalgorithmus).

Schnelle Multipolmethoden

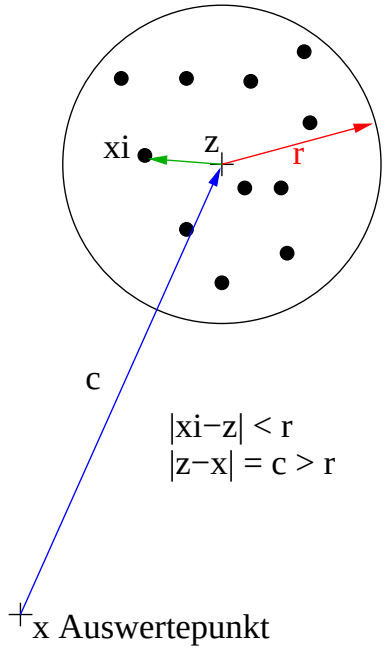
Schnelle Multipolmethoden:

- Die erste grundlegende Idee wurde von *Pincus und Scherega* 1977 veröffentlicht. Wesentliche Idee war die Darstellung einer Gruppe von Partikeln durch ein sogenanntes Pseudopartikel. Dieses repräsentiert die Gruppeneigenschaften und somit das resultierende Potential. Die Beziehung mit einer anderen Partikelgruppe kann dann mit einer einzigen Multipol-Entwicklung errechnet werden.
- Ein zweites Konzept ist die hierarchische Vergröberung des Raumes in separierte Teilgebiete.
- Beide Methoden wurden durch Appel, 1985 und Barnes und Hut, 1986 innerhalb eines Algorithmus zusammengefaßt. Der Aufwand beträgt $O(NlogN)$.
- Die schnelle Multipol Methode wird 1987 von Greengard und Rokhlin veröffentlicht. Zu den beiden genannten Ideen führen sie noch eine lokale Entwicklung von Potentialen ein. In bestimmten Fällen, etwa bei gleichmäßiger Partikelverteilung, reduziert sich der Aufwand auf $O(N)$.

Schnelle Summationsmethoden I

Beschleunigte Methode zur Kraftauswertung:

- Wir betrachten den abgebildeten Cluster aus Körpern: M Massenpunkte seien in einem Kreis um z mit dem Radius r enthalten. Wir werten das Potential ϕ aller Massenpunkte im Punkt x mit $\|z - x\| = c > r$ aus.



- Betrachten wir zunächst einen Massenpunkt an der Position ξ mit $\|\xi - z\| < r$. Das Potential der Masse in ξ sei (der multiplikative Faktor γm wird vernachlässigt).

$$\phi_\xi(x) = \frac{1}{\|\xi - x\|} = f(\xi - x).$$

Schnelle Summationsmethoden II

- Das Potential hängt nur vom Abstand $\xi - x$ ab.
- Nun fügen wir den Punkt z ein und entwickeln in eine Taylorreihe um $(z - x)$ bis zur Ordnung p (nicht mit Prozessor verwechseln):

$$\begin{aligned} f(\xi - x) &= f((z - x) + (\xi - z)) = \\ &= \sum_{|\alpha| \leq p} \frac{D^\alpha f(z - x)}{|\alpha|!} (\xi - z)^{|\alpha|} + \underbrace{\sum_{|\alpha| = p} \frac{D^\alpha f(z - x + \theta(\xi - z))}{|\alpha|!} (\xi - z)^{|\alpha|}}_{\text{Restglied}} \end{aligned}$$

für ein $\theta \in [0, 1]$. Wichtig ist die Separation der Variablen x und ξ.

- Die Größe des Fehlers (Restglied) hängt von p, r und c ab.
- Wie kann die Reihenentwicklung benutzt werden, um die Potentialauswertung zu beschleunigen?
- Dazu nehmen wir an, dass eine Auswertung des Potentials der M Massen an N Punkten zu berechnen ist, was normalerweise $O(MN)$ Operationen erforderlich macht.

Schnelle Summationsmethoden III

- Für die Auswertung des Potentials an der Stelle x berechnen wir

$$\begin{aligned}
\phi(x) &= \sum_{i=1}^{M} \gamma m_i \phi_{\xi_i}(x) = \sum_{i=1}^{M} \gamma m_i f((z-x)+(\xi_i - z)) \approx \\
&\underset{\substack{\text{(Taylorreihe bis} \\ \text{Ordnung } p\text{)}}}{\approx} \sum_{i=1}^{M} \gamma m_i \sum_{|\alpha|\le p} \frac{D^\alpha f(z-x)}{|\alpha|!} (\xi_i - z)^{|\alpha|} = \\
&\underset{\substack{\text{(Summe} \\ \text{vertauschen)}}}{=} \sum_{|\alpha|\le p} \frac{D^\alpha f(z-x)}{|\alpha|!} \underbrace{\left(\sum_{i=1}^{M} \gamma m_i (\xi_i - z)^{|\alpha|} \right)}_{\substack{=: M_\alpha, \\ \text{unabhängig von } x!}} = \\
&= \sum_{|\alpha|\le p} \frac{D^\alpha f(z-x)}{|\alpha|!} M_\alpha
\end{aligned}$$

- Die Berechnung der Koeffizienten M_α erfordert einmalig $O(Mp^3)$ Operationen.
- Sind die M_α bekannt, so kostet eine Auswertung von $\phi(x)$ $O(p^5)$ Operationen.
- Bei Auswertung an N Punkten erhält man also die Gesamtkomplexität $O(Mp^3 + Np^5)$.

Schnelle Summationsmethoden IV

- Es ist klar, dass das so berechnete Potential nicht exakt ist. Der Algorithmus macht nur Sinn, wenn der Fehler so kontrolliert werden kann, dass er vernachlässigbar ist (z.B. kleiner als der Diskretisierungsfehler).
- Ein Kriterium zur Fehlerkontrolle gibt die Fehlerabschätzung:

$$\frac{\phi_\xi(x) - \sum_{|\alpha|\leq p} \frac{D^\alpha f(z-x)}{|\alpha|!}(\xi - z)^{|\alpha|}}{\phi_\xi(x)} \leq c(2h)^{p+1},$$

  mit $\frac{r}{c} \leq h < \frac{1}{4}$. Für den Fall $c > 4r$ reduziert sich der Fehler wie $(1/2)^{p+1}$.
- Die Näherung wird also umso genauer,
  - je kleiner $\frac{r}{c}$
  - je größer der Entwicklungsgrad p.

Gradientenberechnung

- Im N-Körper-Problem will man nicht das Potential, sondern die Kraft, also den Gradient des Potentials, ausrechnen.
- Dies geht mittels

$$\frac{\partial \phi(x)}{\partial x_{(j)}} \overset{\text{Reihenentw.}}{\approx} \frac{\partial}{\partial x_{(j)}} \sum_{|\alpha| \leq p} \frac{D^\alpha f(z-x)}{|\alpha|!} M_\alpha = \sum_{|\alpha| \leq p} \frac{D^\alpha \partial_{x_{(j)}} f(z-x)}{|\alpha|!} M_\alpha$$

$\uparrow$ Raumdimension

- Man muss also nur $D^\alpha \partial_{x_{(j)}} f(z-x)$ statt $D^\alpha f(z-x)$ berechnen.
- Oben haben wir eine Taylorreihe verwendet. Dies ist nicht die einzige Möglichkeit einer Reihenentwicklung. Daneben gibt es für die in Betracht kommenden Potentiale $\frac{1}{\log(\|\xi - x\|)}$ (2D) und $\frac{1}{\|\xi - x\|}$ (3D) andere Entwicklungen, die sogenannten Multipolentwicklungen, die sich besser eignen.
- Für diese Reihen gelten bessere Fehlerabschätzungen in dem Sinne, dass sie die Form

$$\textit{Fehler} \leq \frac{1}{1 - \frac{r}{c}} \left(\frac{r}{c}\right)^{p+1}$$

  haben und somit schon für $c > r$ erfüllt sind.
- Ausserdem ist die Komplexität in Bezug auf p besser (p^2 in 2D, p^4 in 3D).

Gradientenberechnung

- Eine von Physikern häufig verwendete Approximation des Gravitationspotentials ist eine Taylorentwicklung von

$$\phi(x) = \sum_{i=1}^{M} \frac{\gamma\, m_i}{\|(s-x)+(\xi_i - s)\|},$$

  wobei s der *Massenschwerpunkt* der M Massen ist (und nicht ein fiktiver Kreismittelpunkt).
- Die sogenannte Monopolentwicklung lautet

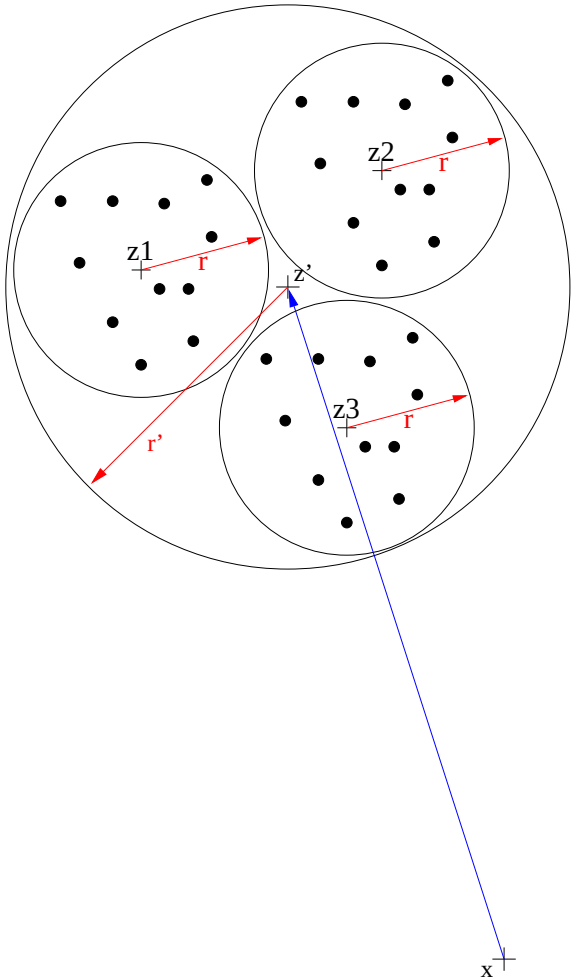
$$\phi(x) \approx \frac{\sum_{i=1}^{M} \gamma\, m_i}{\|s-x\|}$$

  (d.h. ein Körper der Masse $\sum m_i$ in s).
- Die Genauigkeit wird dann nur über das Verhältnis r/c gesteuert.

Verschieben einer Entwicklung

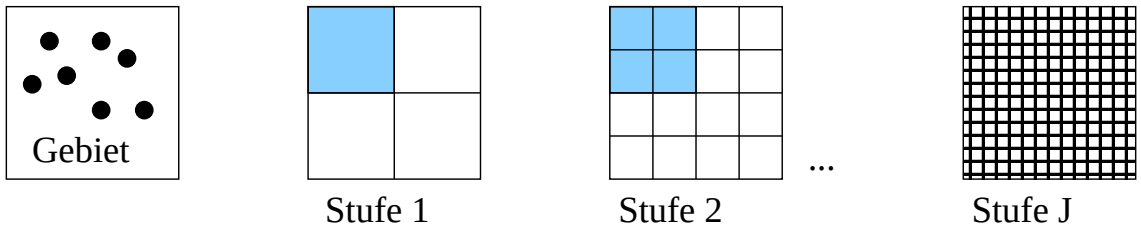
- In den unten folgenden Algorithmen benötigen wir noch ein Werkzeug, das die Verschiebung von Entwicklungen betrifft.
- Die Abbildung zeigt drei Cluster von Körpern in Kreisen um z_1, z_2, z_3 mit jeweils dem Radius r. Die drei Kreise sind in einem größeren Kreis um z' mit Radius r' enthalten.
- Wollen wir das Potential aller Massen in x mit $\|x - z'\| > r'$ auswerten, so könnten wir eine Reihenentwicklung um z' verwenden.
- Falls schon Reihenentwicklungen in den drei kleineren Kreisen berechnet wurden (d.h. die Koeffizienten M_α), so lassen sich die Entwicklungskoeffizienten der neuen Reihe aus denen der alten Reihen mit Aufwand $O(p^\alpha)$ berechnen, d.h. unabhängig von der Zahl der Massen.
- Dabei entsteht *kein* zusätzlicher Fehler, und es gilt auch die Fehlerabschätzung mit entsprechend größerem r'.

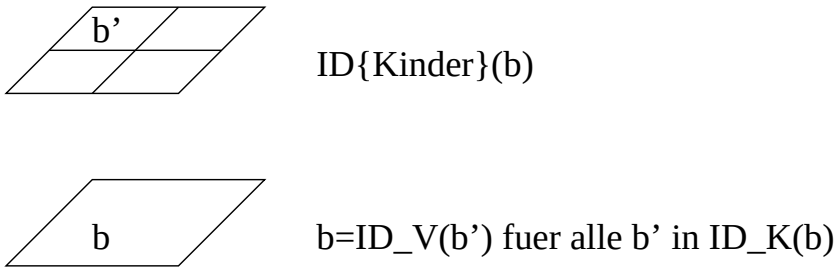
Verschieben einer Entwicklung



Gleichmäßige Punkteverteilung

- Zunächst entwickeln wir einen Algorithmus, der sich für eine uniforme Verteilung der Punkte eignet. Dies hat den Vorteil einfacher Datenstrukturen und der Möglichkeit einfacher Lastverteilung. Wir stellen die Ideen für den zweidimensionalen Fall vor, da sich dies leichter zeichnen lässt. Alles kann jedoch in drei Raumdimensionen analog durchgeführt werden.
- Alle Körper seien in einem Quadrat $\Omega = (0, D_{max})^2$ der Seitenlänge D_{max} enthalten. Wir überziehen Ω mit einer Hierarchie von immer feineren Gittern. Stufe l entsteht aus Stufe $l-1$ duch Vierteln der Elemente.



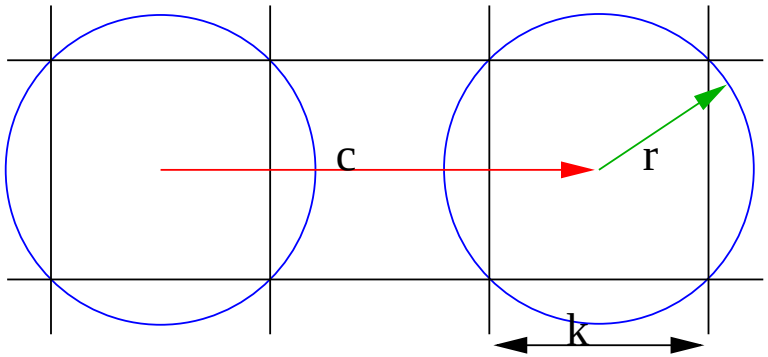


Konstruktion der Gitter

Gleichmäßige Punkteverteilung

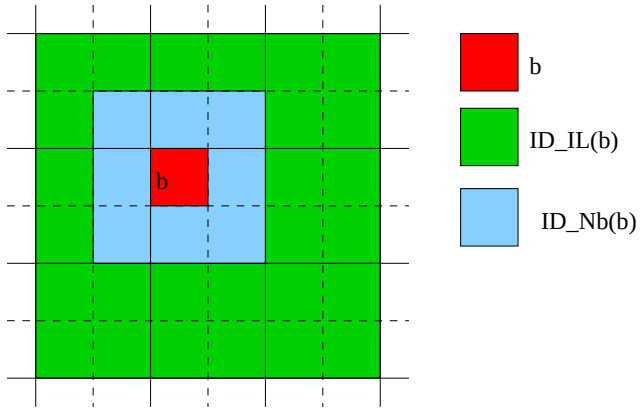
- Wollen wir s Körper pro Feingitterbox, so gilt $J = \log_4\left(\frac{N}{s}\right)$. Für zwei nicht benachbarte Quadrate erhalten wir folgende Abschätzung für das r/c-Verhältnis (Massen in b, Auswertung in a)

$$\begin{aligned} r &= \sqrt{2}\frac{k}{2} \\ c &= 2k \\ \Rightarrow \frac{r}{c} &= \frac{\sqrt{2}\,k}{4k} = \frac{\sqrt{2}}{4} \approx 0.35. \end{aligned}$$



r/c-Abschätzung für zwei nicht benachbarte Quadrate

- Für ein Element b auf Stufe l definiert man folgende Bereiche in der Nachbarschaft von b:
  - $Nb(b) =$ alle *Nachbarn* b' von b auf Stufe l ($\partial b \cap \partial b' \neq \emptyset$).
  - $IL(b) =$ *Interaktionsliste* von b: Kinder von Nachbarn von $Vater(b)$, die nicht Nachbar von b sind.



Gleichmäßige Punkteverteilung

Folgender Algorithmus berechnet das Potential an allen Positionen $x_1, \ldots, x_N$:

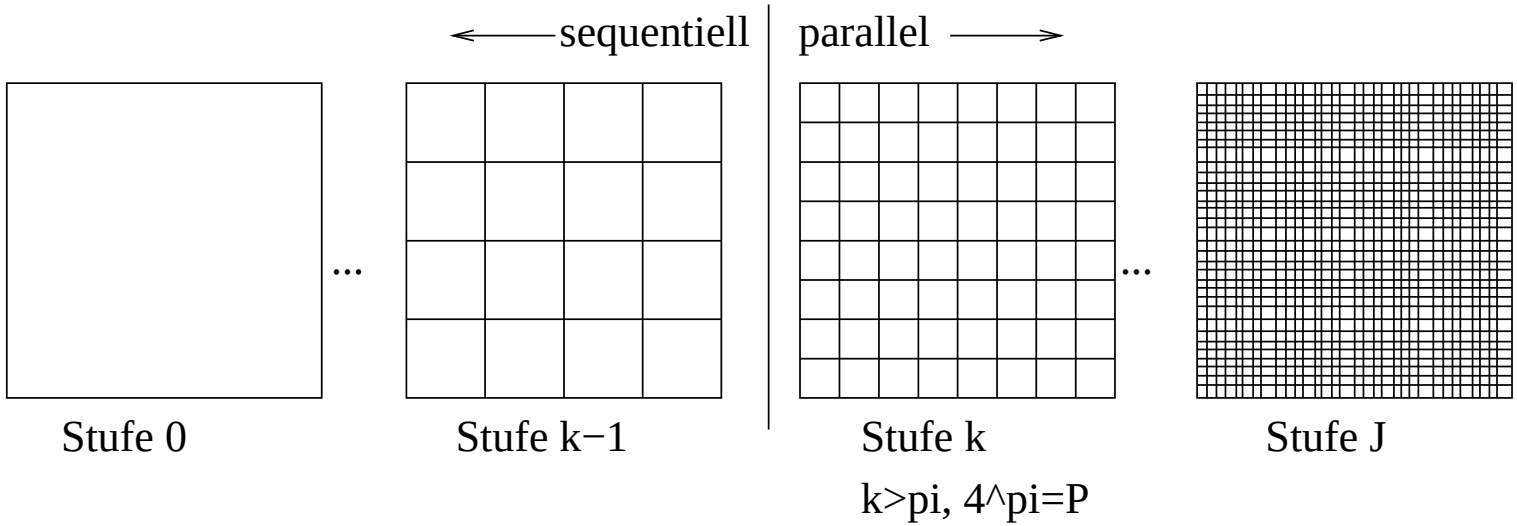
	Aufwand
1. Vorbereitungsphase:	
Für jede Feingitterbox berechne eine Fernfeldentwicklung;	$O(\frac{N}{s} sp^{\alpha})$
Für alle Stufen $l = J-1, \ldots, 0$	
Für jede Box b auf Stufe l	
berechne Entwicklung in b aus Entwicklung in $Kinder(b)$;	$O(\frac{N}{s} sp^{\gamma})$
2. Auswertephase:	
Für jede Feingitterbox b	
Für jeden Körper q in b	
{	
berechne exaktes Potential aller $q' \in B, q' \neq q$;	$O(Ns)$
Für alle $b' \in Nb(b)$	
Für alle $q' \in b'$	
berechne Potential von q' in q;	$O(N8s)$
$\bar{b} = b$;	
Für alle Stufen $l = J, \ldots, 0$	
{	
Für alle $b' \in IL(\bar{b})$	
Werte Fernfeldentwicklung von b' in q aus;	$O(\frac{N}{s} sp^{\beta})$
}	
}	

Gleichmäßige Punkteverteilung

- Gesamtaufwand: $O(N \log N p^\gamma + Ns + Np^\alpha + \frac{N}{s} p^\beta)$, also asymptotisch $O(N \log N)$.
- Dabei bezeichnet α den Exponenten für das Aufstellen der Fernfeldentwicklung und β den Exponenten für das Verschieben.
- Man beachte, dass man wegen der uniformen Verteilung N/s Körper pro Box auf Stufe J hat.
- Die Genauigkeit wird hier über den Entwicklungsgrad p gesteuert, das Verhältnis r/c ist fest.

Parallelisierung: Lastverteilung

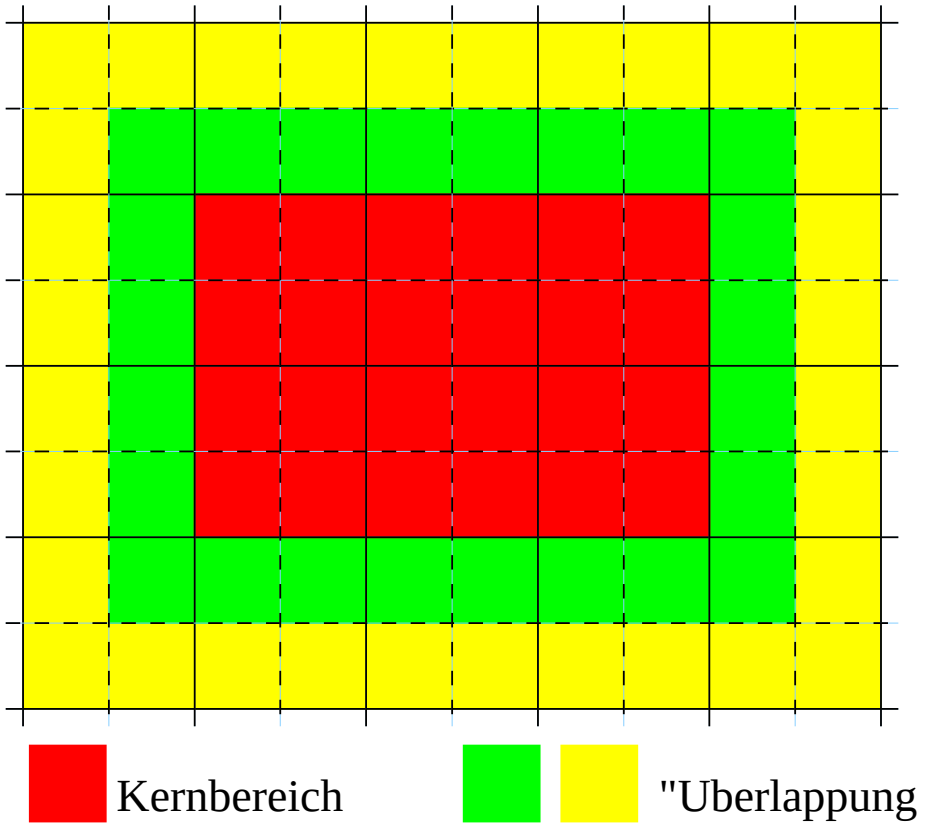
- Lastverteilung:
  Das Gitter mit den zugehörigen Körpern wird auf die Prozessoren aufgeteilt.
- Da wir nun eine Hierarchie von Gittern haben, verfahren wir wie folgt:
  - Jeder Prozessor soll mindestens 2×2 Gitterzellen bekommen.
  - Es sei $P = 4^{\pi}$, so wähle $k = \pi + 1$ und verteile das Gitter der Stufe k auf alle Prozessoren (jeder hat 2×2 Elemente).
  - Alle Stufen $l < k$ werden auf allen Prozessoren gespeichert.
  - Alle Stufen $l > k$ werden so aufgeteilt, dass $Kinder(b)$ im selben Prozessor wie b bearbeitet werden.
  - Beispiel für $P = 16 = 4^2$.



Aufteilung der Boxen bei der Parallelisierung

Parallelisierung: Überlappung

- Zusätzlich zu den ihm zugewiesenen Elementen speichert jeder Prozessor noch einen Überlappungsbereich von zwei Elementreihen:



Überlappungsbereich eines Prozessors

Parallelisierung: Analyse I

- Da jeder mindestens 2×2 Elemente hat, liegt der Überlappungsbereich immer in direkten Nachbarprozessoren.
- Die Auswertung der *IL* erfordert höchstens Kommunikation mit nächsten Nachbarn.
- Zum Aufbau der Fernfeldentwicklung für die Stufen $l < k$ ist eine alle-an-alle Kommunikation erforderlich.
- Skalierbarkeitsabschätzung: Es sei $\frac{N}{sP} \gg 1$. Wegen

$$J = \log_4\left(\frac{N}{s}\right) = \log_4\left(\frac{N}{sP}P\right) = \underbrace{\log_4\left(\frac{N}{sP}\right)}_{J_p} + \underbrace{\log_4 P}_{J_s}$$

werden J_s Stufen sequentiell und $J_p = O(1)$ Stufen parallel gerechnet.

Parallelisierung: Analyse II

- Somit erhalten wir für *festen Entwicklungsgrad:*

$$T_P(N,P) \overset{(\frac{N}{P}=const.)}{=} \underbrace{c_1 \frac{N}{P}}_{\substack{\text{FFE Stufe } J \ldots K \\ \text{Nahfeld auswerten}}} + \underbrace{c_2 \operatorname{ld} P + c_3 P}_{\substack{\text{alle-an-alle. Es sind} \\ \text{immer Daten für 4 Zellen}}} + \underbrace{c_4 p}_{\substack{\text{berechne FFE in ganz } \Omega \\ \text{für } l = k-1 \ldots 0 \text{ in} \\ \textit{allen Prozessoren}}} + \underbrace{c_5 J_p \frac{N}{P}}_{\text{FFE Stufen } l \geq k} + \underbrace{c_5 \frac{N}{P} J_s}_{\text{FFE Stufen } l < k}$$

- Also:

$$E(N,P) = \frac{c_5 N \log N}{\Big(c_5 \frac{N}{P}\underbrace{(J_s + J_p)}_{\log N} + \underbrace{(c_3 + c_4)P}_{\substack{\text{alle an alle} \\ \text{Grobgitter FFE}}} + \underbrace{c_2 \operatorname{ld} P}_{\substack{\text{alle-an-} \\ \text{alle}}} + \underbrace{c_1 \frac{N}{P}}_{\text{Nahfeld lokale FFE}}\Big)P} =$$

$$= \frac{1}{1 + \frac{c_4+c_4}{c_5} \cdot \frac{P^2}{N \log N} + \frac{c_2}{c_5} \cdot \frac{P \operatorname{ld} P}{N \log N} + \frac{c_1}{c_5} \cdot \frac{1}{\log N}}$$

- Für eine isoeffiziente Skalierung muss N also beinahe wie P^2 wachsen!

Parallelisierung: Ungleichmäßige Verteilung

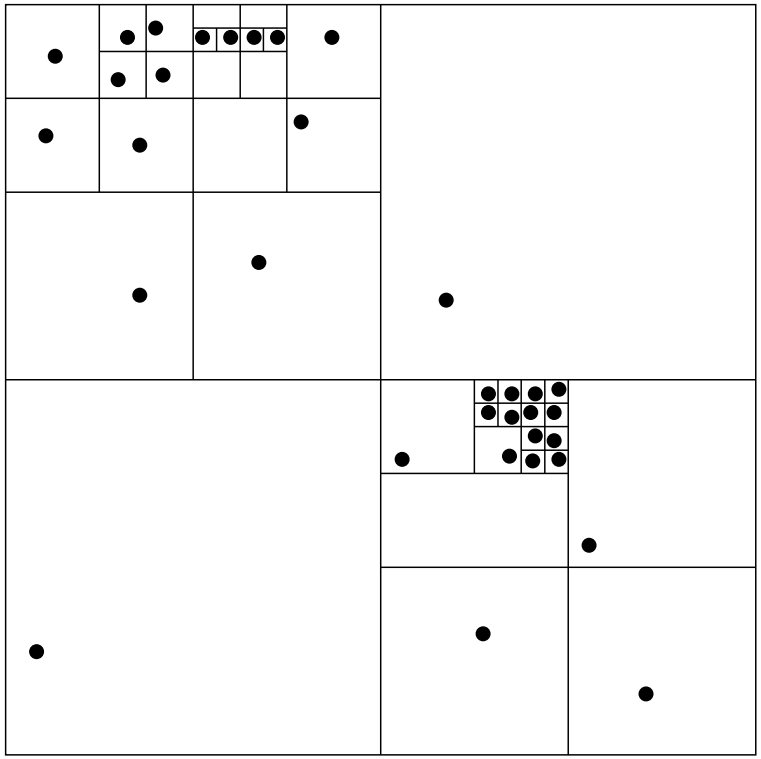
- Die Annahme einer uniformen Verteilung der Körper ist in einigen Anwendungen (z.B. Astrophysik) nicht realistisch.
- Will man in jeder Feingitterbox genau einen Körper haben und ist D_{min} der minimale Abstand zweier Körper, so braucht man ein Gitter mit

$$\log L = \log \frac{D_{max}}{D_{min}}$$

  Gitterstufen. L heisst *„separation ratio“.*
- Von diesen sind aber die meisten leer. Wie bei dünnbesetzten Matrizen vermeidet man nun die leeren Zellen zu speichern. In zwei Raumdimensionen heisst diese Konstruktion „adaptiver Quadtree“.
- Der adaptive Quadtree wird folgendermaßen aufgebaut:
  - Initialisierung: Wurzel enthält alle Körper im Quadrat $(0, D_{max})$.
  - Solange es ein Blatt b mit mehr als zwei Körpern gibt:
    - Unterteile b in vier Teile
    - Packe jeden Körper von b in das entsprechende Kind
    - leere Kinder wieder wegwerfen.

Parallelisierung: Ungleichmäßige Verteilung

- Beispiel eines adaptiven Quadtrees:



- Der Aufwand beträgt (sequentiell) $O(N \log L)$.
- Der erste (erfolgreiche) schnelle Auswertealgorithmus für ungleichmäßig verteilte Körper wurde von Barnes und Hut vorgeschlagen.
- Wie im gleichmäßigen Fall wird eine Fernfeldentwicklung von den Blättern bis zur Wurzel aufgebaut (bei Barnes & Hut: Monopolentwicklung).

Ungleichmäßige Verteilung

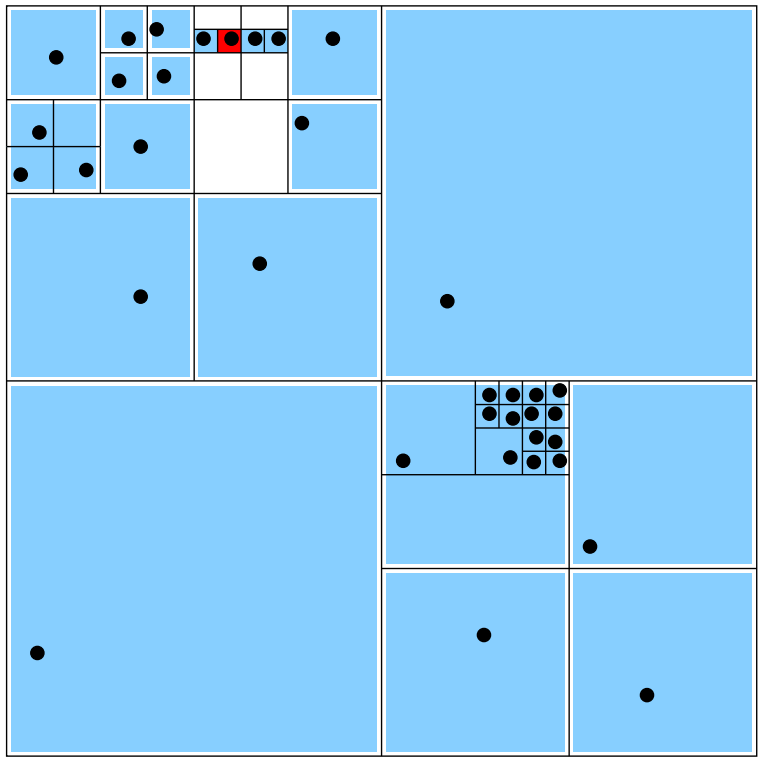
- Für einen Körper q berechnet dann folgender rekursive Algorithmus das Potential in q:

```
Pot(Körper q, Box b)
{
    double pot = 0;
    if (b ist Blatt ∧ q = b.q) return 0;        // Ende
    if (Kinder(b) == ∅)
        return φ(b.q, q);                       // direkte Auswertung
    if (r(b) / dist(q,b) ≤ h)
        return φ_b(q);                          // FFE Auswerten
    for (b' ∈ Kinder(b))
        pot = pot + Pot(q, b');                 // rekursiver Abstieg
    return pot;
}
```

- Zur Berechnung wird *Pot* mit q und der Wurzel des Quadtree aufgerufen.
- Im Algorithmus von Barnes & Hut wird die Genauigkeit der Auswertung mittels des Parameters h gesteuert.

Ungleichmäßige Verteilung

Welche Zellen des Quadtree werden im Barnes & Hut Algorithmus besucht?



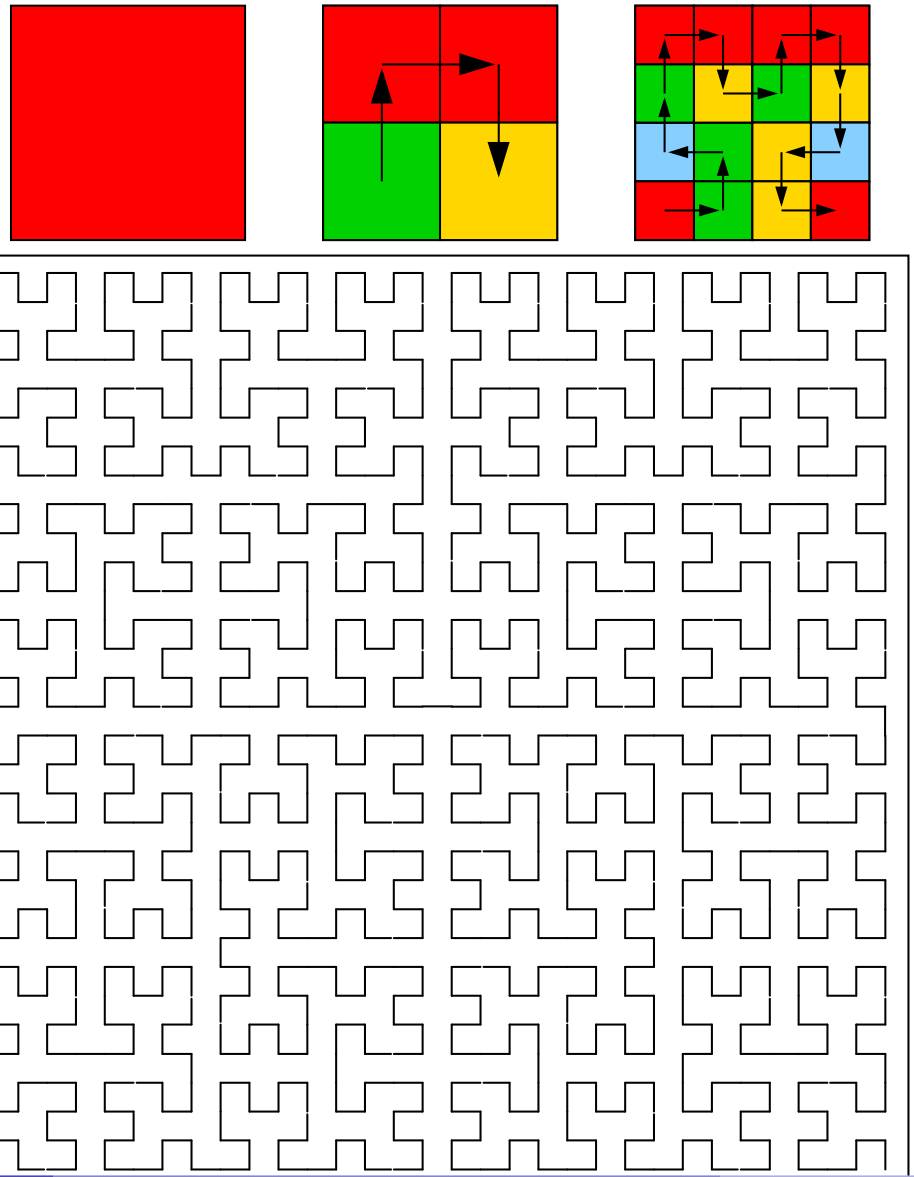
Beispiel zur Auswertung im Barnes & Hut Algorithmus

Ungleichmäßige Verteilung: Parallelisierung I

- Die Parallelisierung dieses Algorithmus ist relativ komplex, so dass wir nur einige Hinweise geben können. Für Details sei auf Salmon, 1994 verwiesen.
- Da sich die Positionen der Teilchen mit der Zeit verändern, muss der adaptive Quadtree in jedem Zeitschritt neu aufgebaut werden. Zudem muss man die Aufteilung der Körper auf die Prozessoren so vornehmen, dass nahe benachbarte Körper auch auf möglichst nahe benachbarten Prozessoren gespeichert sind.

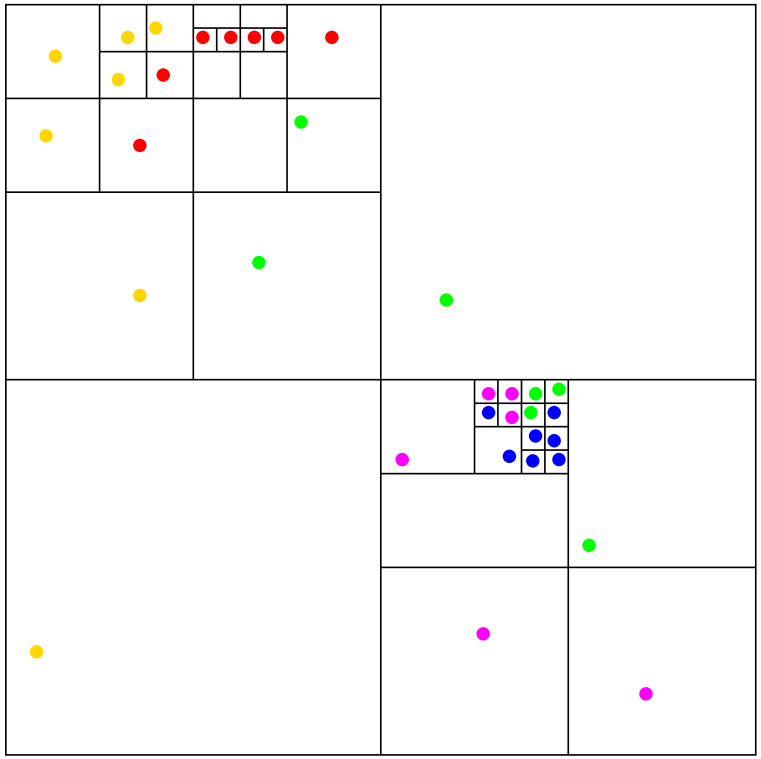
Ungleichmäßige Verteilung: Parallelisierung II

- Ein sehr geschicktes Lastverteilungsverfahren arbeitet mit „raumfüllenden Kurven“. Die sogenannte Peano-Hilbert-Kurve hat folgende Gestalt:



Ungleichmäßige Verteilung: Parallelisierung III

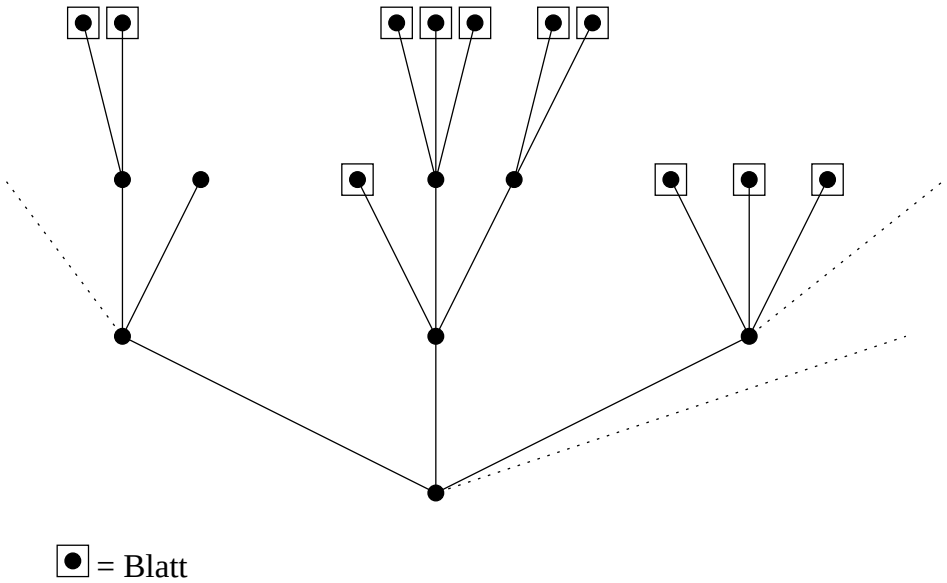
- Eine Hilbertkurve entsprechender Tiefe kann benutzt werden, um eine lineare Ordnung der Körper (bzw. der Blätter im Quadtree) herzustellen. Diese kann dann leicht in P Abschnitte der Länge N/P zerlegt werden.



- Salmon & Warren zeigen, dass mit dieser Datenverteilung der adaptive Quadtree parallel mit sehr wenig Kommunikation aufgebaut werden kann. Ähnlich wie im uniformen Algorithmus wird dann durch eine alle-an-alle Kommunikation die Grobgitterinformation aufgebaut, die alle Prozessoren speichern.

Paralleler Aufbau des adaptiven Quadtree

- *Ausgangspunkt:* Jeder Prozessor hat eine Menge von Körpern, die *einem* zusammenhängenden Abschnitt auf der Hilbertkurve entspricht.
- *Schritt 1:* Jeder Prozessor baut lokal für *seine* Körper den Quadtree auf. Die „Nummern“ der Blätter sind dabei aufsteigend.

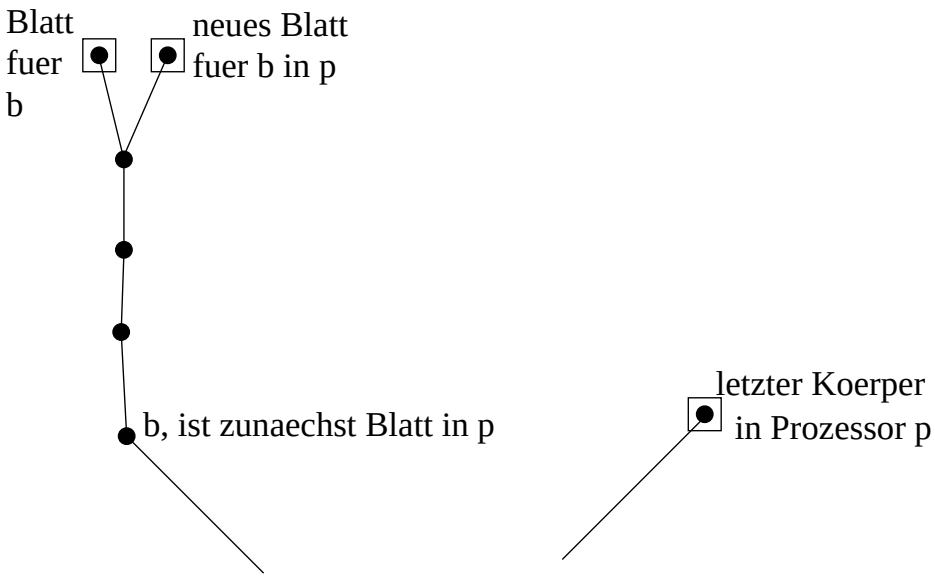


Lokaler Quadtree

- *Schritt 2:* Abgleich mit Nachbarprozessoren. Frage: Hätte ein sequentielles Programm für die Körper von Prozessor p dieselben Blätter erzeugt? Im allgemeinen nein, denn ein Körper von p und einer von $q \neq p$ könnten sich ja eine Box teilen.

Paralleler Aufbau des adaptiven Quadtree

- Was kann passieren?
  Sei b der *erste Körper* in Prozessor p und b' der *letzte* in Prozessor $p-1$. Nun gibt es zwei Möglichkeiten in Prozessor p:
  1. Körper b' liegt in der selben Box wie Körper b. $\Longrightarrow$ Zerteile Box so lange, bis beide Körper getrennt sind. Das neue Blatt ist das selbe, das auch ein sequentielles Programm berechnet hätte! Falls dem *nicht* so wäre, so müsste es einen Körper b'' in Prozessor $q \neq p$ geben, so dass $b'' \in$ neues Blatt von b. Dies ist aber unmöglich, da alle b'' vor b' oder nach dem letzten Knoten von Prozessor p kommen!



Austausch der Randblätter

  2. Körper b' liegt nicht in derselben Box wie Körper b. $\Longrightarrow$ es ist nichts zu tun.
- Für den letzten Körper in p und den ersten in $p+1$ verfährt man ebenso!

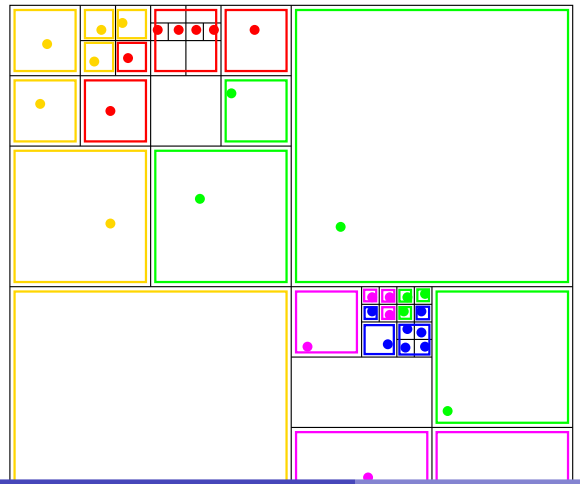
Das Grobgitterproblem

- Wie im uniformen Fall wird der Quadtree von der Wurzel bis zu einer bestimmten Tiefe in jedem Prozessor gespeichert, so dass für diese Fernfeldauswertungen keine Kommunikation notwendig ist oder, falls nötig, der Prozessor bekannt ist, der über die entsprechende Information verfügt.

Definition

Eine Box *b* im Quadtree heisst *Zweigknoten*, falls *b* nur Körper eines Prozessors *p*, der Vater von *b* jedoch Körper verschiedenener Prozessoren enthält. Diesem Prozessor *p* „gehört“ der Zweigknoten.

- Alle Boxen des Quadtree von der Wurzel bis einschließlich der Zweigknoten werden auf allen Prozessoren gespeichert.



Kraftauswertung & Kommunikation

- Ist beim Auswerten ein rekursiver Aufruf in einem Zweigknoten nötig, so wird eine Nachricht an den entsprechenden Prozessor geschickt, dem der Zweigknoten gehört. Dieser bearbeitet die Anfrage asynchron und schickt das Ergebnis zurück.
- Nach dem Update der Positionen berechnet man eine neue Hilbertnummerierung (geht in $\frac{N}{P} \log L$ ohne Aufbau eines Quadtree) für die lokalen Körper. Dann bekommt jeder wieder einen Abschnitt der Länge $\frac{N}{P}$. Dies geht z.B. mit einem parallelen Sortieralgorithmus!
- Salmon & Warren können mit ihrer Implementierung 322 Millionen Körper (!) auf 6800 Prozessoren (!, Intel ASCI-Red) simulieren.