Galaxy Simulation

Jugend Forscht 2018/2019

Emile Hansmaennel
Theodor-Fliedner-Gymnasium, Heinrich Heine Universität Düsseldorf

2018 - 2019

Ist es möglich die Entstehung von Galaxien zu simulieren? Um diese Frage zu beantworten bin ich zu dem Schluss gekommen, dass ich das doch mal ausprobieren sollte. Dazu habe ich das Navarro-Frenk-White Profil implementiert um Cluster an Sternen zu generieren und anschließen die Kräfte die Zwischen den Sternen wirken zu berechnen. Dabei stattete ich die Sterne mit einer zufälligen Masse aus und Unterteilte die Galaxie in dynamisch-große Zellen um die Simulation stark zu beschleunigen. Um die Simulation noch stärker zu optimieren, implementierte ich die Simulation sehr modular um diese auf theoretisch mehreren Tausend Servern gleichzeitig laufen zu lassen. Insgesamt sollte es nun möglich sein einen Zeitschritt in einer Galaxie mit 200 Millionen Sternen in ca. *45 Minuten* statt *1265 Jahren* zu berechnen (Angenommen es werden 1 Millionen Kraftberechnungen pro Sekunde durchgeführt).

Inhaltsverzeichnis

1 Einleitung **1**

2 Vorgehensweise **2**

3 Generieren **2**
3.1 Das Navarro-Frenk-White Profil 2
3.2 Random Sampling 2
3.3 Lookup Tabellen 3
3.4 Beschleunigung der Generierung 3

4 Simulieren **3**
4.1 Die Entstehung von Galaxien 3
4.2 Konzepte 4
4.2.1 Zu lösende Probleme 4
4.2.2 Generierung von Quadtrees und entsprechende Bäume 4
4.2.3 Funktion des Barnes-Hut Algorithmus 4
4.3 Kraft-Berechnungen 4
4.3.1 Die Kraft als Vektor 4
4.3.2 Berechnung der auf einen Stern wirkenden Kraft 5
4.4 Stabile Galaxien 5
4.5 Datenbanken 6
4.5.1 Speichern der Sterne 6
4.5.2 Speichern von Bäumen 6

5 Containerisierung und Modularisierung **6**
5.1 Modularisierung des Generators 6
5.1.1 Generator Modul 6
5.1.2 NFW Modul 6
5.1.3 DB Modul 7
5.2 Modularisierung des Simulators 7
5.2.1 Manager 7
5.2.2 Simulator 7
5.2.3 DB Modul 7
5.3 Sonstige Container 7
5.3.1 Viewer 7
5.3.2 Controller 7
5.3.3 Monitoring 7
5.4 Datenbank Skalierung 8
5.4.1 Sharding (Vertikale bzw. Horizontale Fragmentierung) 8
5.4.2 Caching 8

6 Netzwerk **8**

7 Ergebnisse **9**

8 Quellen **9**

1 Einleitung

Das Projekt ist nach meinem vorletzten Jugend-Forscht Projekt entstanden: Ich habe ein Praktikum in astronomischen Rechen Institut in Heidelberg genutzt, um mit einem Doktoranden[1] das Navarro-Frenk-White Profil, das zum Generieren von Punkt Wolken genutzt wird, zu visualisieren. Anschließend hat sich das Projekt ein bisschen verlaufen, irgendwann beschloss ich jedoch, dass das Projekt weiterzuführen und statt nur statische Galaxien zu generieren, dazu überzugehen die Galaxien zu simulieren, also die Entwicklung einer virtuellen Galaxie zu untersuchen.

Eines der entscheidenden Probleme war die Laufzeit der Simulation. Das Problem, das es zu lösen galt, war die

[1] Tim Tugendhat

ursprüngliche Laufzeit der Simulation von $O(n^2)$ soweit zu minimieren, sodass die Simulation einer "echten" Galaxie in absehbarer Zeit durchführbar ist.

Die Simulation von einem Zeitschritt bei 200 Millionen Sternen würde es erfordern $4 \cdot 10^{16}$ Kräfteberechnungen durchzuführen. Im fall von 1.000.000 Berechnungen pro Sekunde wäre die Berechnung für einen Zeitschritt nach ca. **1267 Jahren** fertig. Durch viele Optimierungen schafft es meine Software die Anzahl an Kräften, die berechnet werden müssen auf (bestenfalls) $2.7 \cdot 10^9$ zu reduzieren und somit eine Laufzeit von ca. **45 minuten** zu erreichen.

2 Vorgehensweise

Wie schon in der Einleitung beschrieben habe ich mehrere Techniken kombiniert, um mein Ziel zu erreichen. Das komplette Projekt lässt sich in mehrere Abschnitte unterteilen: Die Generierung der Punkt Wolke, welche als Galaxie abstrahiert wird und als Basis für weitere Berechnungen genutzt wird, das Einfügen der einzelnen Sterne in einen k-nären Baum und die anschließende Simulation, welche durch Nutzen des Barnes-Hut Algorithmus sehr stark beschleunigt wird.
Um einer optimale Skalierbarkeit zu erreichen wird die Datenbank in mehrere Teile unterteilt. Die Simulation wird ebenfalls auf mehrere Servern durchgeführt, wodurch es möglich ist die Skalierung auf (theoretisch) unendlich vielen Systemen laufen zu lassen.

3 Generieren

Das Generieren der statischen Punkt Wolke, aus der die Galaxie abstrahiert wird ist ein wichtiger Bestandteil des Gesamtprojektes, denn alles baut auf ihr auf. Kurz: um Kräfte zwischen Sternen zu berechnen braucht man erstmal Sterne! Dazu wird das Navarro-Frenk-White Profil verwendet, um eine statische Punktwolke zu generieren.

3.1 Das Navarro-Frenk-White Profil

Das Navarro-Frenk-White Profil (NFW-Profile) [1] ist ein Profil, das genutzt wird, um die Räumliche Massen Verteilung von Sternen zu definieren. Es generiert für einen Stern mit dem Abstand r zum Mittelpunkt der Galaxie eine Wahrscheinlichkeit ρ welche definiert, wie Wahrscheinlich es ist das der Stern mit dem Abstand r existiert:

$$\rho_{NFW}(r) = \frac{1}{\sqrt{2\pi} \cdot \sigma} \cdot \exp\left(\frac{-\phi(r)}{\sigma^2}\right) \tag{1}$$

$$\phi(r) = \frac{4\pi \cdot G \cdot f_0 \cdot R_s^3}{r} \cdot ln\left(1 + \frac{r}{R_s}\right)$$

Es kann nun mithilfe der Random-Sampling Methode (3.2) ermittelt werden, ob ein Stern beibehalten wird oder nicht.

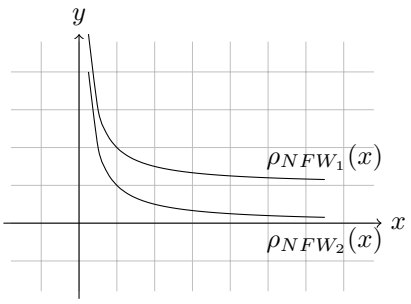


Abbildung 1: Durch Verschiebung des NFW-Profils wird die Annäherung an einen Würfel korrigiert.

Möchte man nun herausfinden wie weit ein Punkt mit der Koordinate (x_1, x_2, x_3) vom Mittelpunkt des Raumes entfernt ist, kann der Satz des Pythagoras (2) verwendet werden.

$$r_3 = \sqrt{x_1^2 + x_2^2 + x_3^2} \tag{2}$$

Der Abstand r_3 zum Mittelpunkt des Raumes kann nun in das NFW-Profil (1) gegeben werden um einen Wert s zu ermitteln, welcher beim Random Sampling verwendet wird:

$$\rho_{NFW}(r) = \cdots = s \tag{3}$$

Dieser Wert s stellt die Wahrscheinlichkeit dar, das ein Stern der eine Entfernung r vom Mittelpunkt der Galaxie besitzt existiert.
Die Galaxie sieht aus der Ferne jetzt jedoch aus wie ein Würfel, da die aus ρ_{NFW_1} resultierende Kurve abrupt endet. Dies kann gelöst werden, indem statt $\rho_{NFW_1}(r)$ folgendes gerechnet wird: $\rho_{NFW_1}(r) - \rho_{NFW_1}(r_{max}) = \rho_{NFW_2}(r)$
Problematisch ist hierbei die Tatsache, dass aufgrund der Verschiebung die Anzahl der Sterne die in Relation zu dem Bereich in dem sie generiert werden sehr stark sinkt und die Zeit um eine bestimmte Anzahl an Sternen zu generieren sehr stark steigt.

3.2 Random Sampling

Um nun zu ermitteln, ob ein Stern beibehalten wird oder nicht, wird die Random-Sampling Methode verwendet. Diese generiert in dem gegebenen Intervall, welches zwischen der minimalen und maximalen Wahrscheinlichkeit, welche aus dem NFW-Profile entnommen werden, einen zufälligen Wert.

$$\psi = [\rho(r_{min}); \rho(r_{max})] \tag{4}$$

Sei s ein zufälliger Wert im Intervall ψ. Generiert man nun einen zufälligen Wert r im Intervall ψ, kann überprüft werden, ob $s > r \vee s < r$ gilt. Ist $r > s$, wird der Stern verworfen, ist $r < s$ wird der Stern behalten.
In der obigen Abbildung ist zu sehen wir zwei zufällige Punkte s_1 und s_2 generiert wurden.
Angenommen es wurde ein Stern generiert für den nach Formel (2) gilt: $r = \sqrt{x_1^2 + x_2^2 + x_3^2} = \cdots = r_1$. Es wird

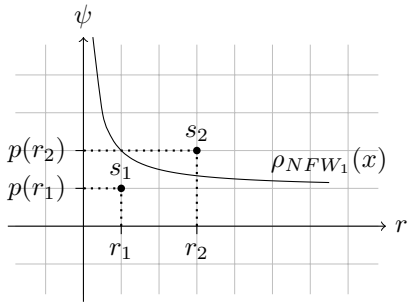


Abbildung 2: Veranschaulichung des Konzeptes, welches beim Random-Sampling verwendet wird.

dann ein Zufälliger Wert $p(r_1)$ im in (4) definierten Intervall generiert. Die folgenden zwei Fälle können dann eintreten und werden wie in (5) beschrieben abgehandelt.

$$\begin{cases} s_1 \leq NFW(r_1) & \rightarrow \text{Stern wird beibehalten} \\ s_1 > NFW(r_1) & \rightarrow \text{Stern wird verworfen} \end{cases} \tag{5}$$

Es kann somit für einen gegebenen Stern geprüft werden ob dieser existiert oder nicht und anhand dessen beschlossen werden, ob der Stern beibehalten wird oder nicht.

3.3 Lookup Tabellen

Statt nun für jeden Stern die Distanz des jeweiligen Sternes r in das NFW-Profil (1) einzusetzen, kann das NFW-Profil im vor hinein berechnet werden. Es wird dabei eine Tabelle erstellt in der die Entfernung des Sternes zum Mittelpunkt der Galaxie der jeweiligen Wahrscheinlichkeit zugeordnet wird:

r_1	ρ_1
r_2	ρ_2
r_3	ρ_3
$\dots$	$\dots$
$r_n \quad n \in \mathbb{N}$	$\rho_n \quad n \in \mathbb{N}$

Die Tabelle kann jedoch nicht so genaue Ergebnisse liefern wie das NFW-Profil, sie kann jedoch so angepasst werden, dass sie in den Arbeitsspeicher passt und somit das NFW-Profil so genau wie möglich widerspiegelt und das Generieren stark verbessert. Dadurch das sie in den Arbeitsspeicher passt, ist der Tatsache das sie nicht direkt berechnet wird auch zu vernachlässigen, denn das Lesen von Daten aus dem Arbeitsspeicher funktioniert sehr schnell. Mit genügend Arbeitsspeicher ist der Fehler demnach auch vernachlässigbar. Ein kritischer Faktor, der beachtet werden muss, wenn Lookuptabellen genutzt werden, ist die Geschwindigkeit des jeweiligen Speichermediums. Nutzt man z. B. Eine sehr langsame Festplatte kann es mehr Sinne machen die jeweiligen Werte direkt zu berechnen. Dagegen ist eine schnelle SSD (Solid-State-Drive) um einiges schneller, kommt jedoch nicht gegen den Arbeitsspeicher an weshalb es sich nicht lohnt ein anders Medium zu nutzen.

3.4 Beschleunigung der Generierung

Es existieren mehrere Möglichkeiten die Generierung der Punkte zu verbessern.

Eine gute Möglichkeit ist die Nutzung von mehr Rechenleistung. Bei der Nutzung von n mal so vielen Rechen Kernen ist das Generieren von Sternen n mal schneller. Die Server des Server-Hosters Hetzner können dabei gut verwendet werden: Es wird stündlich abgerechnet und 32 Kerne mit 128 GB RAM kosten $\approx$ 50ct/h was es ermöglicht für einen Vergleichsweisen Günstigen Preis, sehr viele Koordinaten zu generieren.

Die Ausgabe von jeder Potenziellen Koordinate in die Kommandozeile verlangsamt die Generierung unglaublich stark, da der Rechner darauf wartet, dass die Ausgabe fertig ist bevor er mit der nächsten Rechnung beginnt, was zu einer relativ starken Verlangsamung der Generierung führt. Dies sollte also verhindert werden.

4 Simulieren

4.1 Die Entstehung von Galaxien

"Eine Galaxie ist eine durch Gravitation gebundene große Ansammlung von Sternen, Planetensystemen, Gasnebeln und sonstigen Stellaren Objekten." [2]

Demnach ist es relativ Einfach eine Galaxie zu generieren: es werden einfach ganz viele Objekte in einen Raum geworfen. Das reicht jedoch nicht um die Objekte als Galaxie definieren zu können, da sie nicht "durch Gravitation gebunden" sind.

Um dies zu tun muss die Kraft zwischen allen Objekten in der Galaxie berechnet werden um damit die Position der jeweiligen Objekte nach einer bestimmten Zeit bestimmen zu können.

Tut man dies, indem man zwischen allen Objekten die Kräfte berechnet kommt es zu Problemen: die Anzahl der Kraft Berechnungen die durchgeführt werden müssen steigen exponentiell: Die Anzahl der Kräfte die auf n Sterne wirken lassen sich mit der Formel $n \cdot (n-1)$ berechnen. Für $n = 3$ müssen demnach 6 Kräfte berechnet werden, für 100 Sterne dagegen 9900 und für 1.000.000 Sterne $\approx 9.99999 \cdot 10^{11}$. Die Anzahl der Kraft Berechnungen, die durchgeführt werden müssen beim Simulieren einer "echten" Galaxie mit $> 200 \cdot 10^6$ Sternen ist demnach so groß, dass die Anzahl der Kräfte die berechnet werden müssen minimiert werden müssen, um in einer sinnvollen Zeit an ein Ergebnis zu kommen.

Dies reicht jedoch auch nicht, um eine "stabile" Galaxie zu generieren: berechnet man nur die Kräfte die auf ruhende Objekte in einem Reibungsfreiem Raum wirken, würden alle Objekte zum Massen Mittelpunkt gezogen werden und die Galaxie würde somit implodieren. Es ist also nötig auf die Sterne in der Galaxie Anfangs Kräfte zu wirken. Diese Kräfte sind durch die Rotation der Galaxie um den Massen Mittelpunkt der Galaxie definiert, man rotiert also die Galaxie und gleicht durch die Zentripetalkraft die Kraft die alle Sterne Richtung Massen Mittelpunkt zieht aus. Rotiert man die Galaxie jedoch zu schnell, explodiert sie förmlich, da die Sterne nicht mehr zusammengehalten werden und die Fliehkraft sie einfach auseinanderzieht.

[2] `https://de.wikipedia.org/wiki/Galaxie`

4.2 Konzepte

4.2.1 Zu lösende Probleme

Wie bereits in der Einleitung beschrieben, ist eines der Probleme das Auftritt die Anzahl der nötigen Kraft Berechnungen wodurch der Rechenaufwand quadratisch in Relation zu der Anzahl der Sterne steigt und somit in $O(n \cdot (n-1)) \in O(n^2)$ liegt.
Es kommt ebenfalls zu Problemen, wenn der mittlere Fehler, der bei der Berechnung der Kraft entsteht, größer als die wirkende Kraft wird. Dies passiert unter anderem dann, wenn der Abstand zwischen den Sternen so groß wird, dass die wirkende Kraft so gering ist das sie mithilfe von Computern nicht mehr sinnvoll dargestellt werden kann. Statt nun mit Rundungsfehlern zu rechnen, können diese Sterne, die sehr weit entfernt vom Stern dessen Kräfte berechnet werden sollen, einfach nicht mehr beachtet werden, da sie nicht sinnvoll beitragen. Um diese Sterne jedoch nicht komplett aus der Berechnung auszunehmen, können kleine Cluster an Sternen, welche weit genug vom Stern auf den die Kräfte berechnet werden sollen weg sind und klein genug sind zu einem Pseudo-Stern zusammengefasst werden, welcher durch den Masse Mittelpunkt der Sterne die er repräsentiert definiert ist. Das Konzept wurde 1986 von Josch Barnes und Piet Hut veröffentlicht [2] und erlaubt es die Anzahl an Kräften, die berechnet werden müssen von $O(n^2)$ auf $O(nlog(n))$ zu reduzieren.

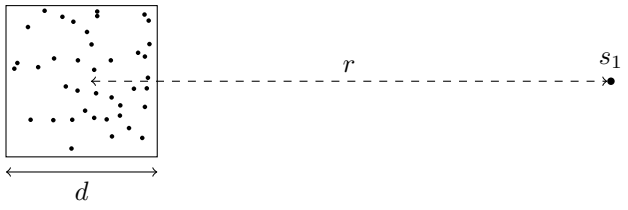
4.2.2 Generierung von Quadtrees und entsprechende Bäume

Um Sterne clustern zu können muss die Galaxie in der sich die Sterne befinden in Zellen unterteilt werden. Dazu wird ein Quadtree bzw. Octree (ein k-närer Baum mit 4 bzw. 8 Kindern) aufgebaut, indem die Sterne eingefügt werden. Damit die Stern-cluster gebildet werden können müssen die Sterne sich in den Blättern des Baumes befinden. Die Knoten des Baumes, indem sich die Sterne befinden dürfen somit keine weiteren Kinder besitzen.
Ein Galaxie wie in Abbildung 4 dargestellt wird demnach Stern für Stern in einen anfangs leeren Baum eingefügt. Die Größe der ersten Zelle die nach dem einfügen aller Sterne alle Sterne beinhaltet kann falls bekannt ist in was für einem Intervall die Koordinaten der Sterne sich befinden direkt genutzt werden, um die Größe der Zelle zu definieren, andernfalls muss diese erst ermittelt werden, indem die minimal und maximal Koordinaten in der liste an Sternen gesucht werden. Beim Einfügen kann es jedoch wie in Abbildung 6 zu sehen zu Problemen kommen. Im falle, dass der Knoten, indem ein Stern eingefügt werden soll bereits durch einen anderen Stern belegt ist müssen für den jeweiligen Knoten Kinder-Knoten erzeugt werden in den die beiden Sterne eingefügt werden müssen. Dies wird so lange fortgeführt bis sich alle Knoten in den Blättern des Baums befinden.
Eine Zelle ist durch ihren Mittelpunkt und ihrer Breite definiert, daher verschiebt sich eine Zelle, welche sich eine Ebene "tiefer" befinden um ein Viertel ihre Breite und die Breite der neuen Zelle ist im Relation zu der ursprünglichen Zelle gesehen halbiert.

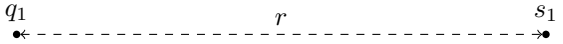
Nachdem alle Sterne in einen Baum eingefügt wurden und sich in den Blättern des Baumes befinden, wird der Baum rekursiv durchlaufen und es wird für jeden inneren Knoten (Knoten, welche keine Blätter sind) die gesamt Masse ihrer Kinder berechnet und der Massen Mittelpunkt.

4.2.3 Funktion des Barnes-Hut Algorithmus

Um nun zu definierten, welche Cluster zusammengefasst werden und welche nicht wird der Barnes-Hut Algorithmus (6) verwendet. Dieser berechnet einen Wert θ welcher als Referenz dafür genommen werden kann, wie die Relation zwischen der Entfernung zu einem Cluster und der Größe des Clusters ist. Um nun Cluster welche weit genug von einem Stern weg sind und gleichzeitig klein genug sind zu erkennen und herauszufiltern kann der Algorithmus in Kombination mit einem vordefinierten Grenzwert genutzt werden.



(a) Ein Cluster auf mehreren Sternen



(b) Die abstraction des obigen Clusters

Abbildung 3: Visuelle Darstellung der Funktionsweise des Barnes-Hut Algorithmus. Das Stern Cluster aus 3a wird in 3b zu einem Stern abstrahiert.

$$\theta = \frac{d}{r} \tag{6}$$

Ist das Verhältnis zwischen Entfernung r und Breite b des Clusters kleiner als der vorher definierte Grenzwert, kann das Cluster zu einem Pseudostern zusammengefasst werden.

4.3 Kraft-Berechnungen

4.3.1 Die Kraft als Vektor

Um die Kraft als Vektor zu berechnen, welcher ein Stern B auf einen Stern A ausübt, wird die folgende Formel verwendet:

$$\vec{F}_{AB} = \underbrace{-G\frac{m_A m_B}{|r_{AB}|^2}}_{Scalar} \cdot \underbrace{\frac{r_B - r_A}{|r_B - r_A|}}_{Vector} \tag{7}$$

Die Summe der Kräfte, die auf einen Stern wirken ist somit die Summe aller Kräfte die zwischen dem jeweiligen Stern a und allen anderen Sternen wirken:

$$F_a = \sum_{\substack{i=0 \\ i \neq a}}^{n-1} F_{ai} \tag{8}$$

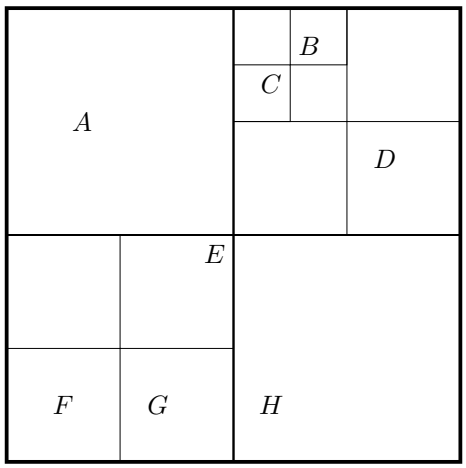


Abbildung 4: Unterteilung einer Galaxie in verschiedene Zellen

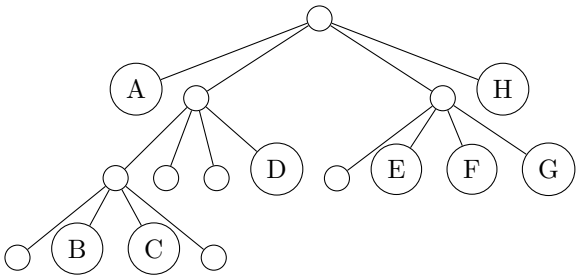
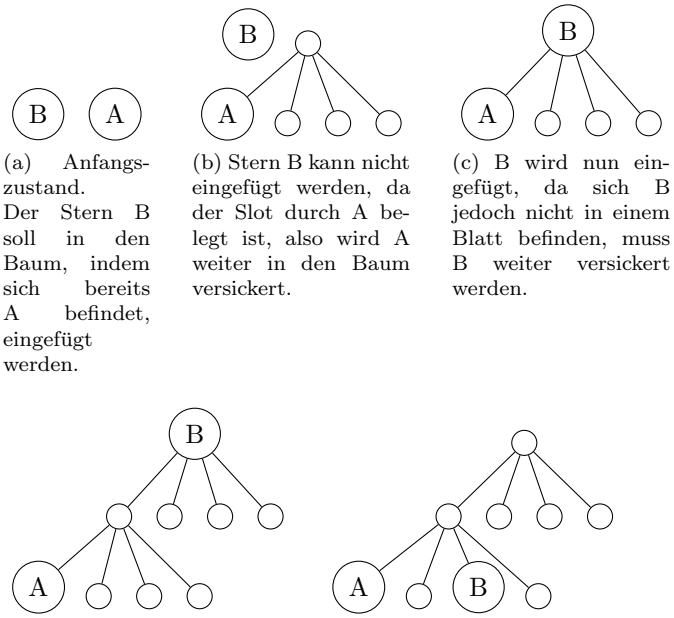


Abbildung 5: Die in Abbildung 4 dargestellte Galaxie als Baum dargestellt

4.3.2 Berechnung der auf einen Stern wirkenden Kraft

Um die Kraft, welche auf einen bestimmten Stern wirkt, zu berechnen, wird der Baum von der Wurzel aus rekursiv durchlaufen. Es wird für jeden Knoten das jeweilige θ berechnet und mit dem vorher definierten Grenzwert verglichen. Ist das berechnete θ kleiner als der vorher definierte Grenzwert wird die Rekursion nicht weiter in den Baum gehen, sondern den jeweiligen Teilbaum zusammenfassen. Es ist somit durch den Grenzwert somit eine End-Bedingung gegeben, welche verhindert das zu weit in den Baum vorgedrungen wird und somit auch verhindert, dass Sterne die in einer zu großen Entfernung zu des Ursprungs Stern liegen und dicht genug gruppiert sind in die Berechnung miteinbezogen werden.
Möchte man die Kraft auf den Stern F in Abbildung 4 berechnen berechnet man das θ zwischen F und dem Wurzel-Knoten. Ist das berechnete θ größer als der vorher definierte Grenzwert kann statt weiter in den Baum hineinzugehen und weitere Kräfte zu berechnen die Kraft zwischen F und dem Pseudostern, der durch den Wurzel-knoten dargestellt wird direkt berechnet werden. Andernfalls wird weiter in den Baum hineingegangen und auf der nächsten Baum-Ebene das Entsprechende θ zwischen den Sternen berechnet werden.
Betrachtet man Abbildung 4 fällt auf das die Zelle in der sich die Sterne C und D befinden sehr klein ist und die Entfernung zu dem Stern F vergleichsweise hoch ist. Es kann also angenommen werden, dass das Theta zwischen F und der Zelle in der sich C und B befinden sehr klein ist. Die Sterne können demnach zusammengefasst werden.



(a) Anfangszustand. Der Stern B soll in den Baum, indem sich bereits A befindet, eingefügt werden.

(b) Stern B kann nicht eingefügt werden, da der Slot durch A belegt ist, also wird A weiter in den Baum versickert.

(c) B wird nun eingefügt, da sich B jedoch nicht in einem Blatt befinden, muss B weiter versickert werden.

(d) Damit B versickert werden kann, wird der Platz der durch A besetzt wird freigemacht, indem A weiter versickert wird.

(e) B kann jetzt in den Baum versickert werden und ist nun ein Blatt.

Abbildung 6: Schrittweises einfügen des Sternes B in einen Baum, indem sich bereits ein Stern (A) befindet.

4.4 Stabile Galaxien

Um eine stabile Galaxie zu generieren, muss diese rotiert werden damit die Zentrifugalkraft der Kraft, welche die Sterne zum Massemittelpunkt zieht entgegenwirkt. Dies lässt sich einfach berechnen, jedoch entsteht ein großes Problem: Es muss eine Kraft ermittelt werden, welche am Anfang der Simulation wirkt.

Da das Ziel meines Projektes nicht darin liegt das Anfangswertproblem zu untersuchen, sondern die restliche Simulation stark zu parallelisieren habe ich beschlossen das Problem in ein eigenes, einfach austauschbares Modul zu verpacken und so in der Zukunft einfach austauschbar zu machen.
Um jedoch eine "halb" Stabile Galaxie zu generieren rotierte ich einfach (im zwei Dimenionalem) die Kraft die im ersten Zeitschritt auf den Stern wirkte um 90 Grad. Dadurch rotieren die Sterne anfangs um den Massenmittelpunkt. Dieses Konzept ist in keinem Sinne physikalisch korrekt, es kann jedoch, als Platzhalter genutzt werden bis einer richtige Möglichkeit gefunden wird dies zu berechnen.

4.5 Datenbanken

4.5.1 Speichern der Sterne

Die Sterne werden in einer Tabelle in Datenbank PostgreSQL gespeichert. Die Tabelle ist wie in Abbildung 7 zu sehen aufgebaut.

star_id	x	y	vx	vy	m
1	-300	300	0	0	1000
2	-200	-200	0	0	1000
...	...	...	...	...	...
n	x_n	y_n	vx_n	vy_n	m_n

Abbildung 7: Darstellung der Tabelle in der die Sterne gespeichert werden. Die star_id Spalte beinhaltet eine global einmalige ID wodurch jeder Stern identifiziert werden kann.

Dadurch das jeder Stern eine einmalige ID besitzt kann diese verwendet werden, um einfach auf Sterne zu verweisen. Dies ist im Kontext des Einfügens sehr hilfreich, da die Verschiebung eines Sternes durch Ändern der Stern-ID vollzogen werden kann.
Jeder Stern besitzt eine Position, einer Geschwindigkeit und eine Masse. Dadurch kann man einen Stern definieren, jedoch auch berechnen was für eine Kraft der Stern auf andere Sterne auswirkt.

4.5.2 Speichern von Bäumen

Um die Bäume in denen die Galaxien gespeichert werden in einer Datenbank zu speichern, muss eine einheitliche Struktur definiert werden, um Probleme in der Zukunft zu verhindern. Die Nutzung von speziellen Graphen Datenbanken bietet sich natürlich an, jedoch wird diese starke Spezialisierung schnell zu einem Hindernis. Um nach dem KISS Prinzip[3] eine möglichst einfache Lösung zu nutzen werden die Bäume in einer Relationalen Datenbank gespeichert. Jeder Knoten wird dabei in einer Zeile der Datenbank gespeichert und erhält eine global einzigartige ID. Die Kinder in andrem Knoten hängen werden anhand ihrer ID in der Zeile gespeichert, sodass es einfach möglich ist einfach auf diese zuzugreifen und beim rekursiven durchsuchen des Baumes auf diese zuzugreifen. Möchte man einen Teilbaum unterteilen können einfach vier neue Knoten erzeugt werden, welche vom Knoten an dem sie hängen referenziert werden.

5 Containerisierung und Modularisierung

Um eine optimale Skalierbarkeit zu erreichen wird die Anwendung in einzelne Module aufgeteilt und in einzelne Container verpackt. Dadurch ist es einfach möglich die Anwendung auf mehreren Rechnern gleichzeitig laufen zu lassen und entsprechende Interaktionen zwischen den Container zu definieren. Durch die containerisierung mithilfe von Docker[4] ist es ebenfalls möglich die Container auf einem beliebigem Betriebssystem zu starten, ohne von bestimmten Bibliotheks-Versionen abhängig zu sein, da diese in den Container bereits integriert sind.

5.1 Modularisierung des Generators

Um den Generator zu modularisieren muss erst definiert werden was für potenzielle Module existieren und wie diese miteinander interagieren.
Insgesamt generiert der Generator zufällige Werte in einem gegebenen Intervall, testet mithilfe des NFW-profils ob diese Sterne existieren oder nicht und schreibt die Sterne anschließend in eine Datenbank. Es sind sofort ein paar Module ersichtlich: ein Modul, welches die Zufälligen Koordinaten generiert, ein Modul, welches den Wert aus dem NFW-Profil berechnet und ein Modul, welches die Daten in die Datenbank schreibt.

5.1.1 Generator Modul

Das Generator Modul generiert zufällige Koordinaten in einem definiertem Intervall und sendet diese an einen NFW Container. Damit nicht ein Container unter der last der einkommenden Antworten leidet, wird der Reverse-Proxy[5] Traefik[6] verwendet. Dieser routet die Anfragen an weitere Container weiter wodurch einer optimale Lastverteilung und Skalierbarkeit gegeben ist.

5.1.2 NFW Modul

Das NFW-modul erhält einen Wert und berechnet den entsprechenden NFW Wert. Dadurch das er durch Traefik angesteuert wird kann, falls die Anzahl der Anfragen zu hoch wird, einfach ein identische Container gestartet werden. Traefik erkennt diesen Container automatisch und kann diesen beim Routen der Anfragen entsprechend nutzen.

[3] "Das KISS-Prinzip (englisch Keep it simple, stupid) fordert, zu einem Problem eine möglichst einfache Lösung anzustreben."`https://de.wikipedia.org/wiki/KISS-Prinzip`

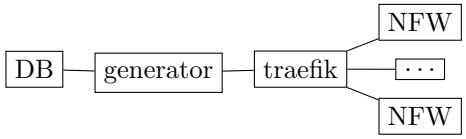
[4] `https://www.docker.com/`

[5] `https://de.wikipedia.org/wiki/Reverse_Proxy`

[6] `https://traefik.io/`

node_id bigint	box_width numeric	total_mass numeric	depth numeric	star_id bigint	root_id bigint	isleaf boolean	box_center numeric[]	center_of_mass numeric[]	subnodes numeric[]
2921847	1000	2000	0	0	1	False	{0, 0}	{0, 0}	{ ... }
2921848	500	1000	1	1	1	True	{-500, 500}	{-300, 300}	{ ... }
2921849	500	0	1	0	1	True	{500, 500}	{0, 0}	{ ... }
2921850	500	1000	1	2	1	True	{-500, -500}	{-200, -200}	{ ... }
2921851	500	0	1	0	1	True	{500, -500}	{0, 0}	{ ... }

Abbildung 8: Darstellung der Tabelle in der ein Baum definiert ist, welcher einmal unterteilt wurde.

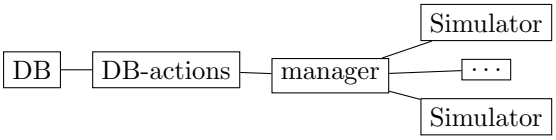


5.1.3 DB Modul

Um die Daten zurück in die Datenbank zu schreiben wird das DB-Modul, welches unter 5.2.3 genau beschrieben wird verwendet.

5.2 Modularisierung des Simulators

Der Simulator simuliert die Sterne aus der Datenbank, indem er Stern für Stern die Kraft, die auf einen Stern wirkt berechnet, die neue Position des Sternes ausrechnet und anschließend den "neuen" Stern zurück in die Datenbank schreibt.



5.2.1 Manager

Um die Simulations-Container optimal skalieren zu können, wird statt den Simulations-Container aktiv Sterne zu geben darauf gewartet, dass ein Simulations-Container einen Stern anfragt. Der Manager fragt im Vorhinein die Datenbank an, um eine Liste an Stern-IDs zu bekommen, auf die die Kraft berechnet werden müssen. Diese Liste an Stern-IDs wird in einen Channel geschrieben, welcher die Sterne einzeln ausgeben kann. Sobald der Channel leer ist, entnimmt der Manager der Datenbank die nächsten Stern-IDs.

5.2.2 Simulator

Der Simulator Container entnimmt dem Manager Container einen Stern und berechnet die Kraft, die auf ihn wirkt, indem er den in der Datenbank gespeicherten Baum in Kombination mit dem Barnes-Hut Algorithmus nutzt. Nachdem die Kraft berechnet wurde kann die neue Position des Sternes berechnet werden und wieder in die Datenbank eingefügt werden.

5.2.3 DB Modul

Der Datenbank Container interagiert mit der Datenbank und stellt verschiedene Methoden zur Verfügung um z. B. Sterne in die Datenbank einzufügen, Daten aus der Datenbank zu erhalten und den Massen Mittelpunkt aller inneren Knoten zu berechnen.
Das Modul stellt ebenfalls Funktionen zur Verfügung welche Sterne die in die Funktion gegeben werden in die Datenbank einfügen und lesen können.

5.3 Sonstige Container

5.3.1 Viewer

Um sich das Endergebnis anschauen zu können, müssen die Daten aus der Datenbank in ein entsprechendes Format gebracht werden damit sie betrachtet werden können. Dazu nutzt der Viewer-Container die Daten aus der Datenbank und generiert daraus entsprechend Bilder, Videos oder Vektorgrafiken. Die generierten Bilder sind meist in einer sehr hohen Auflösung von 15360x15360px ausgegeben. Problematisch wird hierbei die Datei-Größe: Ein solch großes Bild ist schnell mehrere Hundert Megabytes groß. Um das Problem zu lösen, können die resultierenden Bildern anstatt als Rastergrafik als Vektorgrafik exportiert werden. Dadurch kann die Größe der Datei um ein mehrfaches reduziert werden und es treten keine Effekte wie Unschärfe auf, da die Grafik lokal gerendert wird.

5.3.2 Controller

Der Controller steuert den gesamt Zustand, er bestimmt also was getan werden muss, z. B. wie viele Sterne generiert werden, wo sich die einzelnen Container befinden und wie die Last auf den Container ist.

5.3.3 Monitoring

Um einen Überblick über die Gesamtsituation zu bekommen ist es nicht hilfreich sich auf allen Servern anzumelden und dort nachzugucken wie die Auslastung gerade ist. Um dies an einer Stelle zu "monitoren" verwende ich die "time series database" Prometheus[7] als backend für das Monitoring System Grafana[8].
Die einzelnen Simulations-Container senden alle paar Sekunden die Anzahl der Sterne die sie bereits simuliert haben an einen Manager-Container. Dieser stellt Prometheus

[7] `https://prometheus.io/`
[8] `https://grafana.com/`

wiederum die gesammelten Daten zur Verfügung. Prometheus sammelt die Daten alle paar Sekunden ein und speichert diese um anschließend einen Verlauf in der Form eines Graphen o. ä. darzustellen. Um alle Server zu Monitoren kann Grafana auf mehrere Prometheus Instanzen zugreifen und entsprechende Graphen generieren. Somit ist es möglich mit geringem Aufwand alle laufenden Dienste auf einen Blick zu überwachen. Der gesamte Prozess ist in Abbildung 9 dargestellt.

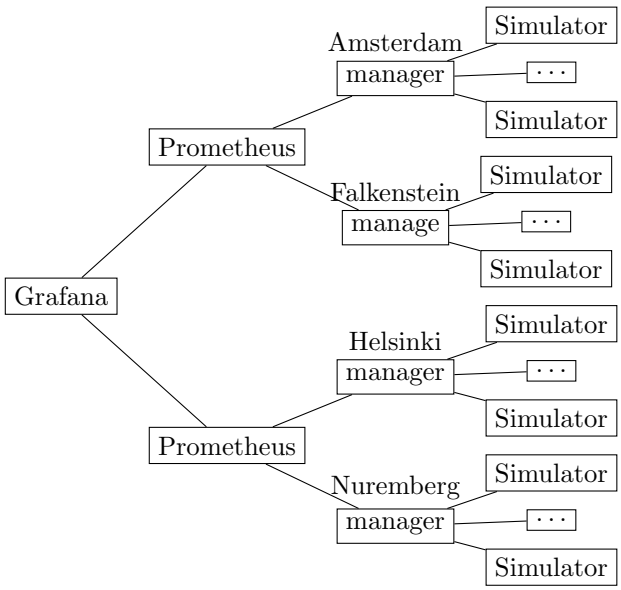


Abbildung 9: Das Monitoren von mehreren Containern

5.4 Datenbank Skalierung

Ein Flaschenhals, der bei der Skalierung entsteht ist die Anbindung an die Datenbank: Desto mehr Simulations-Container mit der Datenbank interagieren, desto höher wird die Auslastung der Datenbank. Um dieses Problem zu lösen, bietet es sich an die Datenbank in mehrere teile aufzuspalten. Ein weiters essenzielles Problem das bei der Verteilung der Simulation rechen Knoten in verschiedene Rechenzentren entsteht ist, dass die Bandbreite zur Datenbank sinkt und die Latenz steigt. Dieses Problem wird durch sogenanntes "Sharding" (vertikale bzw. horizontale Fragmentierung[9]) gelöst.

5.4.1 Sharding (Vertikale bzw. Horizontale Fragmentierung)

Ab einer gewissen Größe kann eine Galaxie nicht mehr in einer Datenbank gespeichert werden. Diese muss demnach auf mehrere Rechner aufgeteilt werden. Da die Datenbank einerseits die einzelnen Sterne Speicher und Bäume, welche die Sterne referenziert bietet es sich hier an diese beiden Bestandteile der Datenbank in einzelne Datenbanken auszulagern.

Sterne (Horizontale Fragmentierung)
Möchte man eine Liste an Sternen auf mehrere Datenbanken aufspalten wird die Liste entsprechend aufgeteilt und auf die Datenbanken verteilt. Wird nun ein bestimmter Stern gesucht, wird die Anfrage über einen reverse-proxy geleitet, welcher die Anfrage an die entsprechende Datenbank weiterleitet.
Es ist somit möglich die Liste an Sternen auf mehrere Datenbanken aufzuteilen und somit die Last von einem System auf mehrere zu verteilen.

Bäume (Vertikale Fragmentierung)
Die Aufteilung der Datenbank in der die Bäume gespeichert werden gestaltet sich ähnlich. Statt alle Bäume in einer Datenbank zu speichern, werden die entsprechenden Teilbäume ab einer bestimmten Tiefe in verschiedene Datenbanken verteilt. Die Interaktion mit der Datenbank verändert sich nur minimal. Statt bei einer Anfrage an die Wurzel eines Baumes die entsprechenden node_ids der Kinder zu bekommen, erhält man die Adresse der Datenbank in der der Teilbaum gespeichert wird. Dies ist in Abbildung 10 visuell dargestellt.

5.4.2 Caching

Ein weiters Problem das mit der Nutzung eines verteilten Systems entsteht ist die Bandbreite zwischen den Simulatoren und der Datenbank und die entsprechende Latenz. Der Durchschnitt mehrerer Messungen zwischen verschiedenen Servern ist in Abbildung 11 dargestellt.
Es wird deutlich, dass falls der Server auf der die Datenbank läuft, sich physisch sehr weit von den Servern, auf denen sich die Simulation-container befinden steht, die Bandbreite zu Problemen führt. Es bietet sich also an die Daten die viel von den Simulations-Containern genutzt werden physisch näher an die Simulations-Container zu bringen um so Probleme die durch die niedrige Bandbreite und hohe Latenz entstehen zu minimieren. Dies ist in Abbildung 12 zu sehen: Es existiert eine Haupt-Datenbank, welche alle Zeitschritte speichert und mit den lokalen Datenbanken kommuniziert. Diese Speichern den jeweiligen Zeitschritt, den die Simulations-Container benötigen, um die Kraft Berechnung durchzuführen. Sobald der Lokale Cache leer ist, wird der Nächste Zeitschritt von der Datenbank in den Cache kopiert und die Simulations-Container können mit der Arbeit fortfahren.
Der Server-Hoster Hetzner bietet eine simple und schnelle Art Caching einfach aufzubauen. Es ist möglich ein virtuelles Volumen zu erzeugen auf das mehrere Rechner in einem Rechenzentrum zugreifen können. Das Virtuelle Volumen wird in die entsprechenden Server als Volumen ein gehangen und ist somit für alle nutzbar. Damit lässt sich an den jeweiligen Standorten einfach ein cache implementieren.

6 Netzwerk

Damit die Container untereinander interagieren können kommunizieren sie über verschiedene APIs. Die NFW-container exponieren eine HTTP API welche einen Wert r annehmen und den jeweiligen NFW Wert ausrechnen.

[9] https://de.wikipedia.org/wiki/Denormalisierung#Fragmentierung

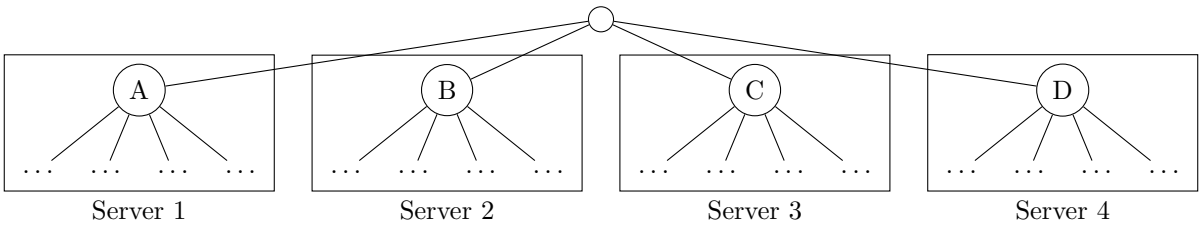


Abbildung 10: Die Teilbäume A, B, C und D werden auf verschiedenen Servern gespeichert und entsprechend angesprochen.

Server 1 (Standort)	Server 1 (Standort)	Bandbreite (Mb/s)	Latenz (ms)
Nuremberg	Helsinki	450	23
Nuremberg	Nuremberg	2000	
localhost	localhost	55000	0.07

Abbildung 11: Messungen der Anbindungen zwischen verschiedenen Servern. Die Messung Nuremberg $\leftrightarrow$ Nuremberg bezieht sich auf zwei Server im gleichen Rechenzentrum und die Messung localhost $\leftrightarrow$ localhost auf die selbe Maschine

7 Ergebnisse

Die Generierung der Punktwolken ist komplett skaliert, es ist nun möglich mehrere Generator-Instanzen hochzufahren welche die Sterne generieren und in eine Datenbank schreiben. Die Sterne in der Datenbank können auch simuliert werden, die "ursprüngliche" Laufzeit in $O(n^2)$ ist auf $O(n \cdot log_4(n))$ reduziert was es (in der Theorie) ermöglicht eine "echte" Galaxie mit 200.000.000 Sternen in **45 Minuten** statt **1267 Jahren** zu simulieren (Faktor 14.808.695). Es wird dabei davon ausgegangen, dass pro Sekunde die Kraft die auf 1.000.000 Sterne wirkt berechnet werden kann. Dies ist auf einem einzlnem Rechner (derzeitig) nicht durchführbar, durch die Aufteilung auf mehrere Rechner ist dies jedoch möglich.
Durch die Aufteitung in mehrere Module welche containerisiert sind, kann die Simulation mithilfe von Crouwdsourcing (in der theorie) durch mehrere freiwillige helfer einfach genutzt werden, wodurch es möglich wird viele personen in das projekt mit einzubeziehen, jedoch nicht zu überlasten.

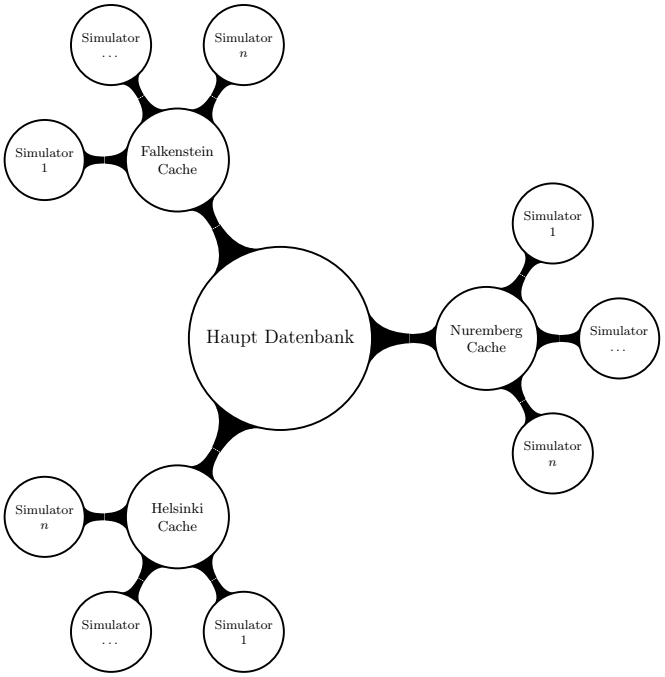


Abbildung 12: Aufspaltung der Datenbank und Nutzung von lokalen Caches

8 Quellen

Literatur

[1] Julio F. Navarro, Carlos S. Frenk and Simon D.M. White, *The Structure of Cold Dark Matter Halos*, 1986.

[2] Josh Barnes and Piet Hut, *A Hierarchical $O(NlogN)$ Force calculation Algorithm*, Nature, vol. 324, pp.446, 1986.

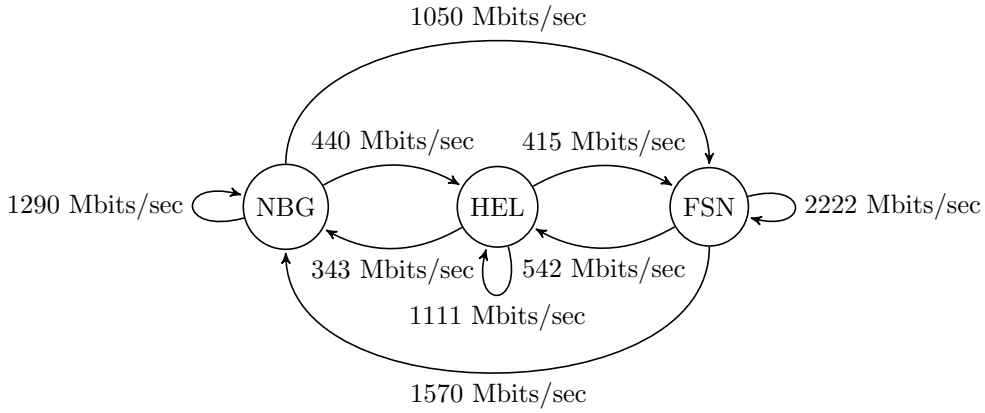


Abbildung 13: Bandbreite zwischen den Rechenzentren des Serverhosters Hetzner (NBG ⇒ Nuremberg, HEL ⇒ Helsinki, FSN ⇒ Falkenstein)