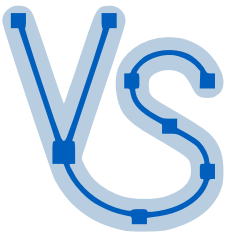
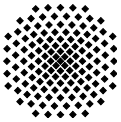
Institut für
Parallele und Verteilte Systeme

Abteilung für Verteilte Systeme





Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 2731

# Optimierung multidimensionaler Bereichsanfragen mittels raumfüllender Kurven in Peer-to-Peer-Netzen

Daniel Tiebler

| | |
|---|---|
| **Studiengang:** | Diplom Informatik |
| **Prüfer:** | Prof. Dr. rer. nat. Dr. h. c. Rothermel |
| **Betreuer:** | MSc. Faraz Ahmed Memon |
| **begonnen am:** | 2008–01–21 |
| **beendet am:** | 2008–08–04 |
| **CR-Klassifikation:** | C.2.4, E.1, H.2.4, H.3.4 |

# Kurzfassung

In dieser Arbeit wird eine Optimierung von multidimensionalen Bereichsanfragen in Peer-to-Peer-Netzen erarbeitet, die auf raumfüllenden Kurven und verteilten Hash-Tabellen basiert. Bisherige Ansätze verwenden entweder nur ein oder alle Attribute der Daten für eine Indexstruktur. Die Optimierung besteht darin, individuelle Attributskombinationen zu erstellen und für Anfragen einen optimalen Index auszuwählen. Die Bildung von Attributskombinationen wird mithilfe einer Heuristik durchgeführt und für die Auswahl einer optimalen Indexstruktur wird ein heuristischer Algorithmus vorgestellt. Zudem werden zwei Optimierungen eingeführt, die die Anzahl der parallelen Nachrichten im Netz begrenzen sowie aufwändige Berechnungen im Netz verteilen.

---

**Optimising Multi-dimensional Range Queries Using Space-filling Curves in Peer-to-Peer Overlay Networks**

# Abstract

The main focus of this thesis is the optimization of multi-attribute range queries on DHT-based peer-to-peer overlay networks, using space-filling curves. Existing approaches either use one or all data attributes to built an index structure. In both cases, the performance of the queries suffers if an application has large number of data attributes. This work optimizes the multi-attribute range queries by building multiple indices on certain attribute combinations and choosing an optimal one for query processing. Additionally two forwarding optimizations are introduced that limit the number of parallel messages in the network and distribute computation tasks uniformly.

# Danksagungen

Zuallererst geht mein Dank an meine Eltern, *Christine* und *Horst Tiebler*, da sie mir die Freiheit ließen, mich dem Studium der Informatik zu widmen, sowie mir über all die Jahre eine Unterkunft und Verpflegung boten. Zudem schenkten sie mir meinen ersten Computer, einen Amiga 500, mit dem ich nicht nur spielen durfte, sondern auch arbeiten sollte. Dies weckte mein Interesse an sowohl der praktischen als auch, später in Verbindung mit dem Informatikunterricht in der Schule, theoretischen Seite dieses Fachgebietes und stellte die Weichen für meinen weiteren Werdegang. Hierfür sei ihnen ebenfalls herzlichst gedankt.

Für eine durchweg sehr gute Betreuung während der gesamten Bearbeitungszeit bedanke ich mich bei meinem Betreuer *Faraz Ahmed Memon*. Er hatte immer Zeit, Sachverhalte ausführlich zu diskutieren und zeigte reges Interesse am Vorankommen sowie an den Ergebnisse dieser Arbeit. Er war auch bei der Übersetzung ins Englische behilflich.

Mein Dank geht zudem an *Frank Dürr*, welcher sich ebenfalls die Zeit nahm, die hier vorgestellten Konzepte anzuhören und zu begutachten. Auch las er die Ausarbeitung Korrektur und fand so manche verwirrenden Formulierungen.

---

# Acknowledgements

First of all, thanks to my parents, *Christine* und *Horst Tiebler*, since they gave me the freedom to pursue my studies in Computer Science and gave me food and shelter over the past years of my life. In addition to that, they gave me my first computer, an Amiga 500, as a present, with which i was not only allowed to play games, but also had to work. This increased my interest for both the practical and, later in conjunction with Computer Science lessons at school, theoretical aspects of the discipline, which set the course for my future career. Therefore, I am extremely thankful to them.

For a very good all-round support during the whole Diploma thesis duration, many thanks go to my supervisor *Faraz Ahmed Memon*. He always had time to discuss the subject in detail and was very interested in the progress as well as the results of this work. He also was helpful with translating into English.

Many thanks go to *Frank Dürr*, who had the time to listen and to have a close look on the herein presented concepts. Furthermore, he proofread the report and pointed out some confusing statements.

# Inhaltsverzeichnis

# Kapitel 1

# Einleitung

Während des letzten Jahrzehnts wurden Peer-to-Peer-Netze (*peer-to-peer overlay networks*) entwickelt, die architektonisch über dem Internetprotokoll liegen und es Endbenutzern ermöglichen, Daten auf skalierbare, effiziente, flexible und zuverlässige Weise auszutauschen. Solche Netze können sowohl zum Tauschen als auch zum Anbieten unterschiedlichster Ressourcen eingesetzt werden, wie zum Beispiel Rechenzeit, Speicherplatz oder generell Daten. Eine Kategorie der Peer-to-Peer-Netze sind die verteilten Hash-Tabellen (*distributed hash tables*), welche den Daten einen eindeutigen Hash-Wert und damit einen definierten Speicherort im Netz zuweisen. Die ersten Ansätze unterstützen nur Punktanfragen (*point queries*) nach einzelnen Werten kostengünstig. Zudem waren die Anfragen nur auf eine Dimension beschränkt, da der Wertebereich der Hash-Werte selbst eindimensional ist. Bereichsanfragen (*range queries*) würden aufgrund der Hash-Funktion weit auseinander liegende Werte erzeugen, die Zugriffe über große Bereiche des Netzes erforderlich machten. Eine solche Anfrage wäre mit großen Kommunikationskosten verbunden. Aufgrund dieser Beschränkung entstanden ausgefeiltere Ansätze (siehe Ganesan u. a. (2004); Schmidt und Parashar (2003); Andrzejak und Xu (2002); Shu u. a. (2005)), welche die verteilten Hash-Tabellen erweitern und kostengünstige Bereichsanfragen sowohl für einzelne als auch mehrere Attribute (*multi-attribute range queries*) ermöglichen.

Dies wird erreicht, indem als Hash-Funktion eine Indexstruktur über die beschreibenden Eigenschaften der Daten, den Attributen, aufgebaut wird. Datenbankverwaltungssysteme setzen Indices zum Organisieren der Daten auf Speichermedien ein, wodurch sich ein effizienter Zugriff ergibt (siehe Kemper und Eikler (2001)). Die Suche nach einem Datum muss demnach nicht alle Daten einer Datenbank berücksichtigen, was zu einer sehr schlechten Leistungsfähigkeit (*performance*) führte, sondern sie beschränkt sich nur auf den Speicherort, auf den der Index verweist. Folglich beschleunigt ein Index

das Auffinden von Daten und erhöht somit die Leistungsfähigkeit der Suche. Das Konzept der Indexstrukturen wurde auf Netze übertragen, wobei es im Falle der verteilten Hash-Tabellen dezentral eingesetzt wird und es somit keinen einzelnen, fehleranfälligen Punkt (*single point of failure*) im System mehr gibt. Als eine solche Indexstruktur können raumfüllende Kurven (*space-filling curve*) dienen. Raumfüllende Kurven bilden mehrere Dimensionen auf eine Dimension ab. Werden die Attribute der Daten als Dimensionen betrachtet, erfolgt die Abbildung von mehreren Attributen in eine Dimension, den Index. Der Vorteil dieser Abbildung ist in erster Linie die Zusammenfassung mehrere Attribute zu einem Index, mit dessen Hilfe nur an einem Ort nach den Daten gesucht werden muss, die alle Attributswerte der Suchanfrage aufweisen. Hinzu kommt die Eigenschaft der raumfüllenden Kurven, dass benachbarte Indexwerte von benachbarten Attributswerten aus abgebildet werden. Diese Lokalität bewahrende Eigenschaft ist in Bereichsanfragen von Vorteil. Bereiche der Attribute werden wahrscheinlich auf benachbarte Indices abgebildet, anstatt über den gesamten Wertebereich des Indexes verteilt zu werden, was die Suchkosten noch weiter senkt.

Allerdings weisen Peer-to-Peer-Systeme, die eine solche Indexstruktur verwenden, zwei Nachteile auf. Ersterer ist der Index selbst, welcher über alle Attribute eines Systems erstellt wird. Werden Anfragen durchgeführt, die nur einen Teil der Attribute verwenden, müssen für die restlichen Attribute alle möglichen Werte angenommen werden. Der Index passt folglich nicht optimal zur Anfrage. Dies führt zu einer sehr großen Anzahl von aufzusuchenden Speicherorten. Zweiter ist die raumfüllende Kurve, welche mit steigender Anzahl an Dimensionen zu mehr nicht benachbarter Indices führt und damit die Lokalität bewahrende Eigenschaft kontinuierlich verliert (siehe Ganesan u. a. (2004); Moon u. a. (2001)). Hierdurch nimmt die Entfernung der Speicherorte zu. Beide Nachteile führen zu höheren Kommunikationskosten und damit zu einer schlechteren Leistungsfähigkeit beim Informationsfund (*information discovery*).

Die Aufgabe dieser Diplomarbeit besteht darin, die Leistungsfähigkeit solcher Peer-to-Peer-Netze zu optimieren, die auf einer raumfüllenden Kurve zusammen mit einer verteilte Hash-Tabelle beruhen. Der Lösungsansatz besteht darin, mehrere anfrage-spezifische Indices mit nur einer verteilten Hash-Tabelle zu verwenden, damit für jede Anfrage ein optimaler Index ausgewählt werden kann. Hinzu gesellen sich zwei Optimierungen, die den Einsatz des entwickelten Systems praxistauglich machen. Zum einen wird die maximale Anzahl von parallelen Nachrichten im Netz begrenzt, um einer Überflutung mit Nachrichten vorzubeugen. Zum anderen wird eine verteilte Berechnung der Indices eingeführt, damit ein einzelner Knoten von einer rechenintensiven Anfrage nicht überlastet wird.

Die Ausarbeitung ist wie folgt aufgebaut. Das nachfolgende Kapitel 2 „Grundlagen“ gibt einen tieferen Einblick in die Konzepte, die in dieser Diplomarbeit zur Anwendung kommen. Hierbei handelt es sich um Peer-to-Peer-Netze, verteilte Hash-Tabellen und raumfüllende Kurven. Zuerst wird das Peer-to-Peer-Modell vorgestellt. Anschließend wird auf Funktionsweise und Eigenschaften der verteilten Hash-Tabellen eingegangen. Am Schluss erfolgt die Definition der hier verwendeten raumfüllenden Kurve sowie eine Beschreibung ihrer rekursiven Konstruktion.

In Kapitel 3 „Verwandte Arbeiten“ wird auf Peer-to-Peer-Netze eingegangen, die entweder eine spezielle Netzstruktur aufweisen oder auf verteilten Hash-Tabellen basieren. Die letztgenannten werden danach kategorisiert, wie sie den Index aufbauen.

Die Architektur und das Systemmodell dieser Arbeit werden in Kapitel 4 „Konzepte zur Optimierung“ festgelegt. Nach einem Überblick werden alle Teile schrittweise eingeführt. Während das Kapitel über die Grundlagen die technischen Aspekte vorstellt, zeigt dieses Kapitel das Systemmodell aus der Sicht einer Anwendung. Es wird detailliert beschrieben, wie sowohl Daten im Netz platziert als auch Anfragen für solche Daten abgearbeitet werden. Neben dem grundlegenden Ansatz werden ebenfalls die beiden oben erwähnten Optimierungen aufgezeigt, mit denen eine praktikable Lösung erreicht wird.

Eine Evaluierung des Systemmodells wird mittels Simulationen durchgeführt. Die Ergebnisse werden in Kapitel 5 „Systemevaluation“ diskutiert. Hier werden Indices für unterschiedliche Attributskombinationen miteinander verglichen, damit der Effekt genau passender Indices klar wird. Ferner werden die beiden Optimierungen dem grundlegenden Ansatz gegenübergestellt und damit ihre Notwendigkeit für eine durchführbare Realisierung bestätigt. Am Ende werden die Ergebnisse noch einmal kurz zusammengetragen.

Während das vorletzte Kapitel 6 „Zusammenfassung“ die gesamte Ausarbeitung zusammenfasst, werden im letzten Kapitel 7 „Ausblick“ Aspekte aufgelistet, die nicht Gegenstand dieser Arbeit sind, jedoch einer näheren Untersuchung bedürfen. Hier werden einige Ideen skizziert, die bedacht werden sollten.

# Kapitel 2

# Grundlagen

Die in dieser Arbeit verwendeten Techniken werden in diesem Kapitel eingeführt. Die Reihenfolge orientiert sich an der technischen Voraussetzung der Themen, welche aufeinander aufbauen. Zu allererst werden Peer-to-Peer-Netze vorgestellt. Diese setzen ein bestehendes Rechnernetz bis hin zum Internetprotokoll voraus, auf dem sie basieren und auf das hier nicht näher eingegangen wird (siehe Tanenbaum (2000)). Daraufhin erfolgt eine Definition der verteilten Hash-Tabellen, die eine bestimmte Art von Peer-to-Peer-Netz sind. Die raumfüllenden Kurven hingegen erweitern die verteilten Hash-Tabellen um die Fähigkeit einer Bereichssuche über mehrere Attribute.

## 2.1 Peer-to-Peer-Netze

In den Anfängen des Internets war das Client-Server-Modell vorherrschend. Dabei gibt es den Dienstanbieter, den Server, und den Dienstnutzer, den Client. Die Abbildung 2.1 auf Seite 14 zeigt links eine Veranschaulichung dieses Modells. Diese klare Trennung ist unter anderem auch technisch bedingt. Die Endsysteme des Internets, womit die Endverbraucher gemeint sind, waren über langsame Kommunikationsverbindungen, meist per Modem oder ISDN, an das Netz angebunden und deren Rechner waren damals noch mit leistungsschwachen 486er-Prozessoren oder deren Nachfolgegeneration bestückt. Mit dieser technischen Ausstattung war es praktisch unmöglich einen Dienst im Internet anzubieten, der einer großen Nutzerschar dauerhaft und zuverlässig zur Verfügung stand. Daher wurden spezielle Hochleistungsrechner eingesetzt, die über eine deutlich schnellere und ständige Internetanbindung verfügten. Diese teuren Gerätschaften wurden von Universitäten, Firmen und Organisationen eingesetzt. Dienste wurden auf diesen dedizierten Maschinen, welche als Server bezeichnet werden, zentral im Netz angeboten.

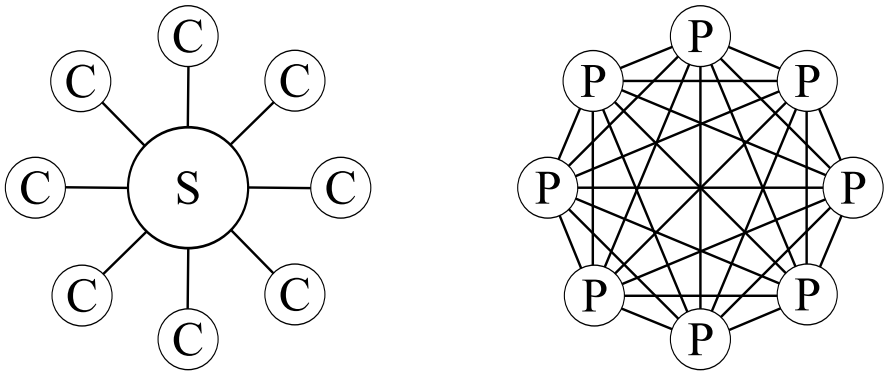



Abbildung 2.1: Die Abbildungen zeigen das Client-Server-Modell (links) sowie das Peer-to-Peer-Modell (rechts).

Mit dem Fortschritt der technischen Entwicklung wurden neben den Internetverbindungen, erste „Flatrates“ sowie DSL-Technik, auch die Computer, deren Prozessortaktfrequenz die Gigahertzgrenze überschritt, immer leistungsfähiger und kostengünstiger. Aufgrund dessen wurde eine direkte Kommunikation zwischen Ebenbürtigen (*Peers*), in diesem Fall den Endverbrauchern, attraktiver. Ein Peer-to-Peer-Netz ist demnach ein Netz, in dem eine direkte Kommunikation von einem Peer zu einem anderen Peer (*Peer-to-Peer*) erfolgt. In Abbildung 2.1 auf Seite 14 sind sowohl ein Server-Client-Modell als auch ein Peer-to-Peer-Modell zum Vergleich skizziert. Anstatt sich eines zentralen Servers als Dienstleister zu bedienen, agieren die Peers selbst als Dienstanbieter gegenüber anderen Peers. Folglich ist im Peer-to-Peer-Modell jeder Peer ein Client und auch gleichzeitig ein Server, womit neue Herausforderungen an die eingesetzten Systeme erwachsen. Zum einen muss ein Peer-to-Peer-System mit einer Vielzahl von Benutzern noch effizient funktionieren, also skalieren. Zum anderen muss es flexibel mit den Bei- und Austritten sowie Ausfall der Benutzer umgehen können, ohne dass der Dienst im Netz zu sehr darunter leidet. Denn im Gegensatz zu dedizierten Maschinen, die ununterbrochen im Netz zur Verfügung stehen, sind Endsysteme nicht ständig „online“. Peer-to-Peer-Netze basieren auf dem Internetprotokoll und werden als ein logisches Netz betrachtet, das ein physikalisches Netz überlagert (*overlay network*). Dabei entspricht eine logische Verbindung im Peer-to-Peer-Netz oft mehreren physikalischen Verbindungen.

Das erste populäre System mit Direktverbindungen zwischen Peers war Napster (Dürr (2007); Mahlmann und Schindelhauer (2007); Saroiu u. a. (2003)), das den Austausch von Dateien (*file sharing*) erlaubt. Es besteht aus einem Server, der als Verzeichnisdienst fungiert. Clients können sich beim

Server anmelden und senden ihm eine Liste mit Dateien zu, die sie selbst bereitstellen. Die Dateilisten sind der Datenbestand des Servers, in dem jeder Client suchen kann. Suchanfragen werden an den Server gesendet, welcher mit einer Liste aller passenden Dateinamen und der Adresse der zugehörigen Clients antwortet. Mit Hilfe der Adressen können die Peers direkt Kontakt aufnehmen und die gewünschte Datei austauschen. Napster ist streng genommen kein reines Peer-to-Peer-Netz, da der Verzeichnisdienst von einem zentralen Server übernommen wird. Allerdings werden die Dateien komplett dezentral gespeichert. Solche Systeme, die sowohl das Client-Server-Modell als auch das Peer-to-Peer-Modell verwenden, werden als hybride Peer-to-Peer-Systeme bezeichnet.

Ein reines Peer-to-Peer-Netz ohne eine zentrale Komponente ist beispielsweise Gnutella (Dürr (2007); Mahlmann und Schindelhauer (2007); Ripeanu (2001); Saroiu u. a. (2003)), das ebenfalls ein Datei-Austausch-Netz ist. Da eine zentrale Anlaufstelle für Suchanfragen fehlt, müssen diese direkt an die anderen Peers gesendet werden, was einer Art Fluten (*flooding*) des Netzes mit Nachrichten gleichkommt. Daher wurden spezielle Suchstrategien entwickelt, die eine komplette Überflutung des Netzes mit Suchanfragen verhindern. Als Beispiele seien die beschränkte Flutung (*limited flooding*) (siehe Lv u. a. (2002)) und der zufällige Weg (*random walk*, siehe Yatin Chawathe and Sylvia Ratnasamy and Lee Breslau and Nick Lanham and Scott Shenker (2003)) genannt. Während beim Erstgenannten die Reichweite der Überflutung begrenzt wird, werden beim Letztgenannten zufällige Wege durch das Netz verwendet. Der Nachteil an solchen System ist, die Daten können sich irgendwo im Netz befinden, da es keinen Zusammenhang zwischen den Daten und der Netzstruktur gibt. Deshalb werden solche Systeme als unstrukturierte Peer-to-Peer-Netze bezeichnet. Es gibt keine Garantie, dass die Daten gefunden werden, wenn sie denn vorhanden sind, außer die Suchanfrage wird an jeden Peer im Netz gesendet.

Eine weitere Kategorie sind die strukturierten Peer-to-Peer-Netze wie die verteilten Hash-Tabellen, die im nachfolgenden Kapitel vorgestellt werden. Diese sind ebenfalls vollständig dezentral. Allerdings besitzen sie einen strukturierten Netzaufbau und Daten werden an wohldefinierten Orten gespeichert. Suchanfragen für bestimmte Daten werden gezielt an den Ort weitergeleitet, wo diese Daten gespeichert sein sollten. Damit ist einerseits eine effizientere Suche als in unstrukturierten Netzen möglich. Andererseits kann garantiert werden, die Daten zu finden, sofern sie im Netz gespeichert wurden.

## 2.2 Verteilte Hash-Tabellen

Verteilte Hash-Tabellen (*distributed hash tables*, siehe Karger u. a. (1997); Stoica u. a. (2001)) sind ein strukturiertes Peer-to-Peer-System. Im Allgemeinen funktionieren sie wie Hash-Tabellen, die in der Programmierung verwendet werden, allerdings ohne den wesentlichen Nachteil der kompletten Reorganisation, wenn sich die Anzahl der Behältnisse ändert.

Daten werden bei Hash-Tabellen als Schlüssel-Werte-Paar angegeben. Der Schlüssel $S$ beschreibt eine eindeutige Kennung für den Wert, wobei der Wert den eigentlichen Daten entspricht oder ein Verweis auf diese ist. Für einen schnellen Zugriff auf die Daten, wird aus dem Schlüssel ein Hash-Wert $H$ berechnet: $h(S) = H$. Die Hash-Tabelle hat $N$ Behältnisse und der Hash-Wert $H$ bestimmt das Behältnis, in dem die Daten gespeichert sind. Damit der Hash-Wert nicht größer als die Anzahl der Behältnisse wird, kommt bei Hash-Funktionen oft eine Modulo-Berechnung zur Anwendung: $h() = (\ldots)\ mod\ N$. Ändert sich die Anzahl der Behältnisse, so müssen die Hash-Werte aller Schlüssel neu berechnet und die Daten in die neu berechneten Behältnisse verschoben werden. Diese Reorganisation ist sehr kostenintensiv. Insbesondere wenn die Behältnisse auf Peers gespeichert werden, da schlimmstenfalls alle Daten über das Netz ausgetauscht werden müssten.

Verteilte Hash-Tabellen reduzieren diesen Aufwand, indem sie ein sogenanntes konsistentes Hash-Verfahren (*consistent hashing*, Karger u. a. (1997); Stoica u. a. (2001)) anwenden. Dabei werden jedem Knoten (Konten, Netzknoten und Peers werden nachfolgend synonym verwendet) und jedem Schlüssel ein $m$-Bit langer Hash-Wert zugewiesen, der Bezeichner (*identifier*) genannt wird. Der Wertebereich des Bezeichners muss ausreichend groß sein, damit eine mögliche Zuweisung des gleichen Bezeichners zu zwei verschiedenen Knoten vernachlässigt werden kann. Die Peers werden in einem „Bezeichner-Kreis“ geordnet, in dem „modulo $2^m$“ gerechnet wird. Der Wertebereich geht folglich von 0 bis $2^m - 1$. Der Bezeichner eines Schlüssels wird dem Peer zugewiesen, dessen Bezeichner größer oder gleich dem Bezeichner des Schlüssels ist. Dieser Peer wird definiert als der *Nachfolger* im Bezeichner-Kreis. In Abbildung 2.2 auf Seite 17 ist eine verteilte Hash-Tabelle zu sehen, wobei die Kreise die Bezeichner der Knoten und die Quadrate die Bezeichner von Daten zeigen. Hierbei ist $m = 6$, weshalb der Bezeichnerbereich von 0 bis 63 geht.

Jeder Peer ist anstelle für einen für mehrere Bezeichner verantwortlich. Ist $B$ die Menge aller Bezeichner sowie $N$ die Anzahl der Peers, so werden jedem Peer mit hoher Wahrscheinlichkeit $\frac{B}{N}$ Bezeichner zugewiesen. Mit hoher Wahrscheinlichkeit meint hier, die Zuweisung erfolgt rein zufällig über die Hash-Funktion und wird damit als beinahe gleichverteilt angenommen.

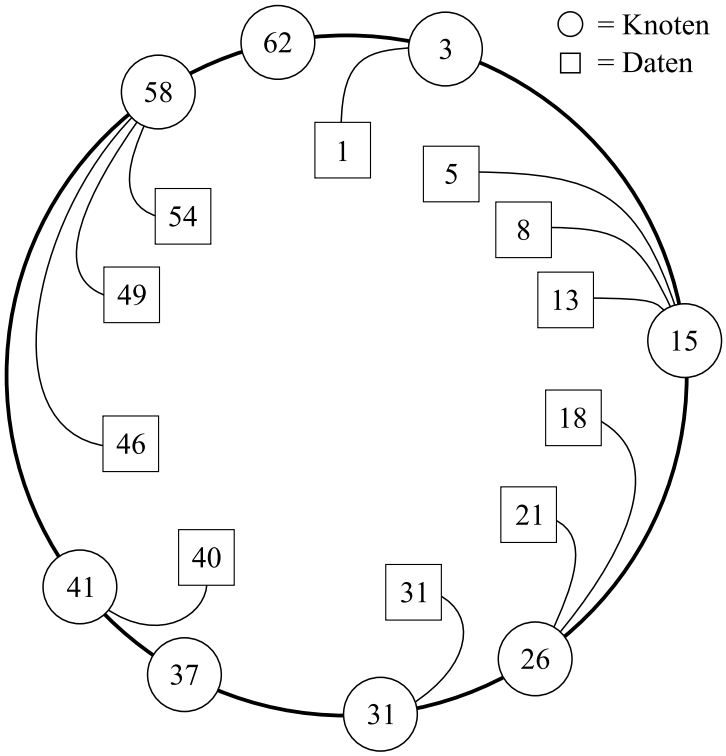


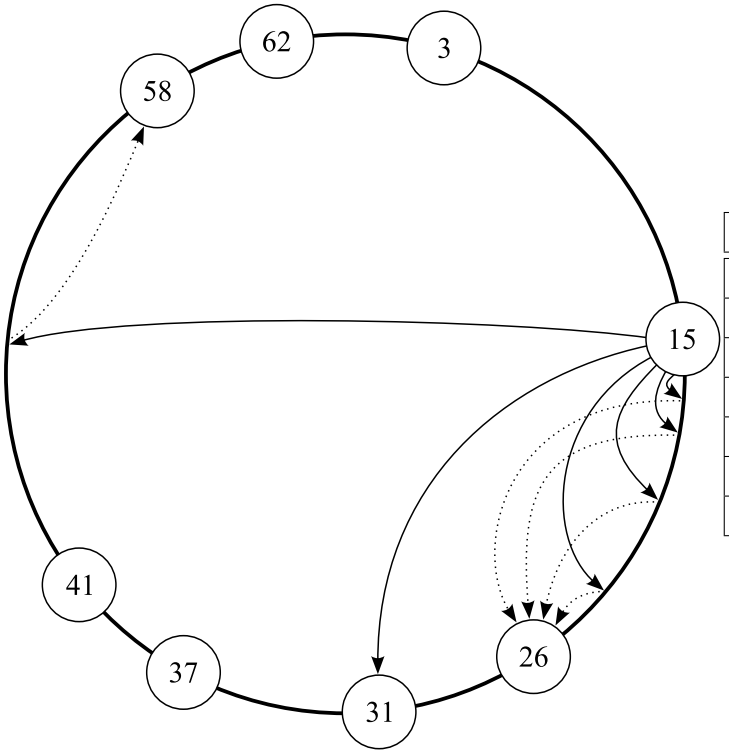

Abbildung 2.2: Zu sehen sind Knoten in einer verteilte Hash-Tabelle mitsamt zugeordneten Daten. Die Zahlen stellen die Bezeichner dar. Mit $m = 6$ ist der Wertebereich des Bezeichners $[0, 63]$.

Hier ist die Hash-Funktion passend zu wählen, welche für gewöhnlich die kryptographische Hash-Funktion SHA-1 (siehe U.S. Dept. Commerce/NIST und National Technical Information Service (1995)) ist und eine sehr gute Zufälligkeit aufweist. Sollte ein Knoten dem Netz beitreten oder es verlassen, müssen mit hoher Wahrscheinlichkeit nur $\mathcal{O}(\frac{B}{N})$ Bezeichner mitsamt ihren Daten verschoben werden. Das betrifft folglich nur all die Bezeichner, für die der entsprechende Knoten verantwortlich ist oder sein wird. Die anderen Zuweisungen bleiben davon unberührt. Dieser Ansatz ist im Vergleich zu einer kompletten Reorganisation deutlich kostengünstiger.

In dieser Arbeit kam das Chord-Protokoll (siehe Stoica u. a. (2001)) zum Einsatz, welches eine verteilte Hash-Tabelle verwendet. Es ist ein skalierbares Peer-to-Peer-Protokoll und kann von Internetanwendungen für eine effiziente Suche verwendet werden. Der Bezeichner-Kreis wird in einem sogenannten Chord-Ring organisiert. Chord bietet zwei wesentliche Funktionen an. Zum einen eine Suchfunktion, mit der effizient der *Nachfolger* eines Bezeichners gesucht werden kann. Zum anderen informiert das Protokoll, sobald sich et-

was an den Bezeichnern ändert, für die ein Knoten zuständig ist. Hiermit kann die Datenmigration initiiert werden. Das Speichern und Verschieben der Daten muss die Anwendung selbst übernehmen. Chord dient einzig und allein der effizienten Suche.

Die Suche im Chord-Ring funktioniert korrekt, wenn jeder Knoten seinen *Nachfolger* kennt. Das ist der Peer, dessen Bezeichner im Ring unmittelbar nach dem Bezeichner eines Knotens kommt. Da ein Knoten seinen *Nachfolger* kennt, weiß er folglich, für welchen Bereich des Rings der nachfolgende Knoten verantwortlich ist. Eine Suchanfrage wird solange im Netz an den *Nachfolger* weitergereicht, bis ein Knoten feststellt, dass der ihm folgende Knoten der *Nachfolger* des gesuchten Bezeichners ist. Dieser Knoten sendet die Adresse des *Nachfolgers* an den Urheber der Suchanfrage, womit letzterer den Speicherort des gesuchten Schlüssel-Werte-Paares gefunden hat.




Finger-Tabelle

| **i** | $\mathbf{n + 2^{i-1}}$ | **ID** |
|---|---|---|
| 1 | 16 | 26 |
| 2 | 17 | 26 |
| 3 | 19 | 26 |
| 4 | 23 | 26 |
| 5 | 31 | 31 |
| 6 | 47 | 58 |

Abbildung 2.3: Neben einem Chord-Ring mit $m = 6$ ist zusätzlich die Finger-Tabelle des Knotens mit dem Bezeichner (*identifier*, ID) 15 zu sehen.

Diese lineare Suche ist jedoch nicht effizient. Deshalb speichert jeder Peer eine sogenannte Finger-Tabelle (*finger table*). Ein Finger ist eine Langstreckenverbindung, zu einem weit entfernten Knoten im Chord-Ring. Eine solche Tabelle enthält höchstens $m$ Finger, deren Bezeichner sich wie folgt berech-

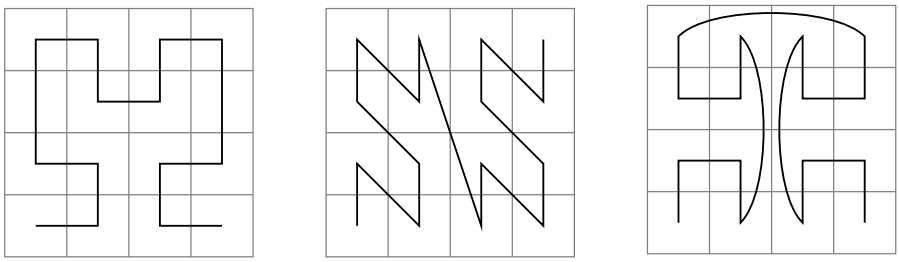
nen. Sei $n$ der Bezeichner des aktuellen Knotens. Dann hat der $i$-te Eintrag den Bezeichner $n + 2^{i-1}$. Zusammen mit dem Bezeichner wird die Adresse des entsprechenden *Nachfolgers* in der Tabelle gespeichert. Der erste Eintrag entspricht dem direkten *Nachfolger*. Bei einer Suche nach dem *Nachfolger* eines Bezeichners wird überprüft, ob einer der Finger im Bereich zwischen dem Bezeichner des aktuellen Knotens und dem zu suchenden Bezeichner liegt. Da mehrere Finger in diesem Bereich liegen können, wird beim größten angefangen. Ist ein Finger gefunden, so handelt es sich um den nahsten Vorgängerknoten, den der aktuelle Knoten kennt. An diesen wird die Suchanfrage weitergeleitet. Wird dieses Verfahren auf jedem Knoten angewendet, so halbiert sich die Entfernung mit hoher Wahrscheinlichkeit zum Zielknoten mit jedem Schritt. Ist $N$ die Anzahl der Knoten, dann werden bei einer Suche mit hoher Wahrscheinlichkeit $\mathcal{O}(\log N)$ Knoten kontaktiert, bis die Suche beendet ist. Damit ist die Suche sehr effizient. Für den Beweis sei auf die Literatur verwiesen. Die Finger-Tabelle ist mit höchstens $m$ Einträgen im Vergleich zu einer maximalen Anzahl von $2^m$ Netzknoten sehr klein, weshalb die Speicheranforderungen des Chord-Protokolls gering ausfallen. Tatsächlich sind in einer Finger-Tabelle Einträge enthalten, die auf die selben Knoten verweisen. Werden diese Mehrfacheinträge nicht gespeichert, reduziert sich der Speicherplatzbedarf der Tabelle auf $\mathcal{O}(\log N)$. In Abbildung 2.3 auf Seite 18 wird ein Chord-Ring sowie eine Finger-Tabelle des Knotens mit dem Bezeichner 15 gezeigt. Wie dort zu sehen ist, könnten die Mehrfacheinträge auf einen Eintrag reduziert werden, die auf den Peer mit dem Bezeichner 26 zeigen.

Der Chord-Ring verhält sich bei ständigem Ein- und Austritt oder sogar bei einem Ausfall von Knoten unter Einsatz weiterer Maßnahmen robust und fehlertolerant. Es kann zwar vorkommen, dass die Finger eines Peers zwischenzeitlich nicht auf den nahsten Vorgänger zeigen, aber die Suche bleibt wegen des direkten *Nachfolgers* weiterhin korrekt. Die Finger-Tabelle wird aufgrund dessen in regelmäßigen Abständen verifiziert und gegebenenfalls aktualisiert. Zudem gibt es keine Einschränkungen in der Wahl des Schlüssels, womit das Chord-Protokoll flexibel einsetzbar ist. Eine nähere Erläuterung der letztgenannten Eigenschaften übersteigt den Rahmen dieser Arbeit, weshalb hier ebenfalls auf die Literatur verwiesen wird.

## 2.3 Raumfüllende Kurven

Nachdem mit dem Chord-Protokoll ein effizientes Peer-to-Peer-Netz für die Suche ausgewählt wurde, bedarf es einer effizienten Indexstruktur für mehrere Attribute, den beschreibenden Eigenschaften der Daten. Eine Indexstruktur wird in Datenbanken dazu verwendet, die Daten auf einem Spei-

chermedium zu organisieren, damit ein Zugriff schnell und effizient erfolgen kann (siehe Kemper und Eikler (2001)). Die Indexstruktur soll sicherstellen, dass möglichst wenige Peers an der Bearbeitung einer Suchanfrage beteiligt sind. Damit werden Kommunikationskosten gespart und die Abarbeitung einer Suchanfrage wird dadurch kostengünstiger und leistungsfähiger. Zum Einsatz kommt eine raumfüllende Kurve, die mehrere Dimensionen auf eine Dimension abbildet. Es existieren verschiedene solcher Kurven wie beispielsweise die Hilbert-, Peano-, Sierpiński-, Moore- oder Gray-Kurven (vergleiche Sagan (1994); Mokbel u. a. (2003)). In Abbildung 2.4 auf Seite 20 werden drei ausgewählte raumfüllende Beispielkurven dargestellt. Diese Diplomarbeit verwendet die Hilbert-Kurve, da sie die beste Lokalität bewahrende Eigenschaft (*locality preserving property*, *clustering property*) aufweist und sie über alle Dimensionen das gleiche Verhalten zeigt (siehe Mokbel u. a. (2003); Moon u. a. (2001); Gotsman und Lindenbaum (1996)).



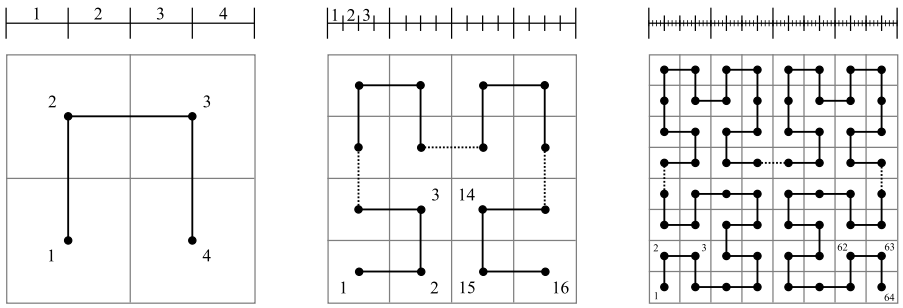
Hilbert-Kurve Peano-Kurve Gray-Kurve

Abbildung 2.4: Es werden die Hilbert-, Peano- sowie Gray-Kurven als Beispiele für raumfüllende Kurven dargestellt.

Aus dem mathematischen Problem, „wie die Punkte einer Linie stetig auf die Punkte eines Flächenstückes abgebildet werden können“ (Hilbert (1891)), entstanden die heute bekannten raumfüllenden Kurven (*space-filling curves*). Obwohl Guiseppe Peano im Jahre 1890 als erster durch arithmetische Betrachtungen dieses Problem gelöst hatte, war es David Hilbert ein Jahr später, der diese Problemstellung geometrisch veranschaulichte. Damit beschrieb er mit einfachen Mitteln, wie Funktionen hergeleitet werden können, die eine solche Kurve abbilden. Im Grunde geht es dabei um die Abbildung $[0,1] \rightarrow [0,1]^2$. Mit den raumfüllenden Kurven kann leicht eine stetige und bijektive Abbildung gezeigt werden, die sich umkehren lässt. Daher ist die für die Indexstruktur benötigte Abbildung von mehreren Dimensionen auf eine Dimension möglich: $[0,1]^2 \rightarrow [0,1]$.

In Abbildung 2.5 auf Seite 21 ist die rekursive Konstruktion der Hilbert-

Kurve zu sehen. Das Flächenstück wird in jeder Dimension in zwei gleichgroße Teile zerlegt und durch die daraus resultierenden vier Teilflächen geht eine Linie. Diese hat die Eigenschaft, jeden Flächenteil nur einmal zu passieren. Die Ziffern geben an, welchem Teilstück auf der Linie welcher Flächenteil entspricht, wobei die Flächen durch ihre Mittelpunkte markiert sind. Wie im Bild zu sehen ist, füllt die Linie die Fläche noch nicht aus. Daher wird mit jeder Teilfläche ebenso verfahren, das heißt jedes einzelne Flächenstück wird wiederum in vier gleichgroße Teile zerlegt. Da die Kurve jedes Teilstück nur einmal durchlaufen darf und stetig sein muss, wird sie aus der vorherigen Kurve konstruiert, damit sie diese Bedingungen erfüllt. Die beiden oberen Teilstücke sind direkte Kopien der ersten Kurve. Die unteren Teilstücke ergeben sich aus einer Drehung um 90° im beziehungsweise gegen den Uhrzeigersinn. Dabei ist auf die Orientierung der Ausgangskurve im zu verfeinernden Flächenstück zu achten. Im dritten Bild wurde auf die gleiche Weise eine weitere Unterteilung durchgeführt. Wie zu sehen ist, nähert sich die Kurve der Fläche an. Je größer die Annäherung, desto mehr Fläche wird durch die Kurve belegt, was bei einer unendlichen Annäherung die gesamte Fläche ergibt. Die arithmetische Konstruktion der Kurve wurde später auf beliebig viele Dimensionen erweitert.




$k = 1$ $k = 2$ $k = 3$

Abbildung 2.5: Es wird die rekursive Konstruktion der Hilbert-Kurve in den ersten drei Annäherungsstufen gezeigt.

Die im Schaubild verwendeten Zahlen werden Hilbert-Bezeichner (*Hilbert identifier*) genannt und sie zeigen, die Abbildung ist nicht auf das Intervall $[0, 1]$ beschränkt, sondern kann auch mit ganzen Zahlen beschrieben werden: $\mathbf{N}^d \rightarrow \mathbf{N}$. Die Anzahl der benötigten Hilbert-Bezeichner kann berechnet werden. Gegeben sei ein Raum mit $d$ Dimensionen. In der ersten Annäherungsstufe (*approximation level*) wird jede Dimension in zwei gleiche Teile zerlegt. Die Anzahl der mit jeder weiteren, rekursiven Raumteilung entste-

henden Unterräume bis zur $k$-ten Annäherungsstufe ergibt sich wie folgt:

$$\begin{array}{lllllllll}
2_1 & \cdot & 2_2 & \cdot & \ldots & \cdot & 2_{d-1} & \cdot & 2_d & = & 2^d \\
2_1^2 & \cdot & 2_2^2 & \cdot & \ldots & \cdot & 2_{d-1}^2 & \cdot & 2_d^2 & = & 2^{2\cdot d} \\
 & & & & & & & & & \vdots & \\
2_1^{k-1} & \cdot & 2_2^{k-1} & \cdot & \ldots & \cdot & 2_{d-1}^{k-1} & \cdot & 2_d^{k-1} & = & 2^{(k-1)\cdot d} \\
2_1^k & \cdot & 2_2^k & \cdot & \ldots & \cdot & 2_{d-1}^k & \cdot & 2_d^k & = & 2^{k\cdot d}.
\end{array}$$

Da alle Unterteilungen einer Dimension in der nächsthöheren Annäherungsstufe zerlegt werden, multipliziert sich deren Anzahl mit zwei. Die Gesamtzahl der Unterräume ergibt sich aus der Multiplikation der Anzahl der Unterteilungen aller Dimensionen. Hiermit lässt sich die Abbildung auch folgendermaßen beschreiben: $[2^k]^d \rightarrow [2^{k\cdot d}]$. Es sei zudem angemerkt, dass an der Berechnungsformel $2^{k\cdot d}$ zu erkennen ist, dass die Anzahl der Unterräume exponentiell mit der Anzahl der Dimensionen und der Annäherungsstufen wächst.

Für eine Indexstruktur werden die Attribute der Daten als die Dimensionen des Raumes der raumfüllenden Kurve betrachtet. Zuerst wird festgelegt, welche Attribute abgebildet werden sollen. Dies bestimmt die Dimensionalität der Hilbert-Kurve. Anschließend sollte geklärt werden, wie hoch die maximale Annäherungsstufe ist. Diese wird durch die „Auflösung“ der Attribute bestimmt. Beträgt der größte Wertebereich aller Dimensionen beispielsweise höchstens 100 diskrete Werte, so ist die maximale Annäherungsstufe: $\lceil \log_2\ 100 \rceil = 7$. Es sei darauf hingewiesen, dass die maximale Annäherungsstufe für alle Dimensionen gleich ist. Ob eine maximale Annäherungsstufe überhaupt Verwendung findet, hängt vom System ab, dass die raumfüllende Kurve einsetzt. Für gewöhnlich wird eine solche definiert und angewendet.

Jener Unterraum, welcher einem Hilbert-Bezeichner zugeordnet ist, wird Zone genannt. Direkt benachbarte Zonen entlang eines Liniensegmentes bilden sogenannte Cluster. Bei einer Punktanfrage, ist höchstens nur ein Abschnitt je Dimension betroffen, sodass nur eine Zone und damit nur ein Hilbert-Bezeichner durch die Abbildungsfunktion berechnet wird. Dies ist der Speicherort, an dem das gesuchte Datum gefunden werden kann. Werden dagegen Bereiche bei der Suche angegeben, so können auch mehrere Abschnitte einer Dimension davon betroffen sein. Eine solche Bereichsanfrage ergibt mehrere Zonen, von denen einige Zonen Cluster bilden. Die Abbildung 2.6 auf Seite 23 veranschaulicht eine beispielhafte Bereichsanfrage. In der vertikalen Dimension ist nur ein Abschnitt und in der horizontalen Dimension sind alle Abschnitte betroffen. Dies führt zu den vier Zonen: 5, 8, 9, 12. Die umittelbar benachbarten Zonen 8 und 9 bilden einen Cluster. Weniger streng genommen bilden die anderen beiden Zonen auch Cluster, aber nur mit einer

Zone. Die Indices beschreiben in diesem Fall mehrere Speicherorte, an denen die Daten gefunden werden können. Dabei ist die Entfernung zwischen den Zonen 5 und 8 sowie 9 und 12 groß. Die Entfernung zwischen den Zonen 8 und 9 ist optimal für ein sequentiell zu lesendes Speichermedium. Kommt eine verteilte Hash-Tabelle zum Einsatz, könnten sich beide Zonen des Clusters auf einem Peer befinden, was die Suche kostengünstiger macht.

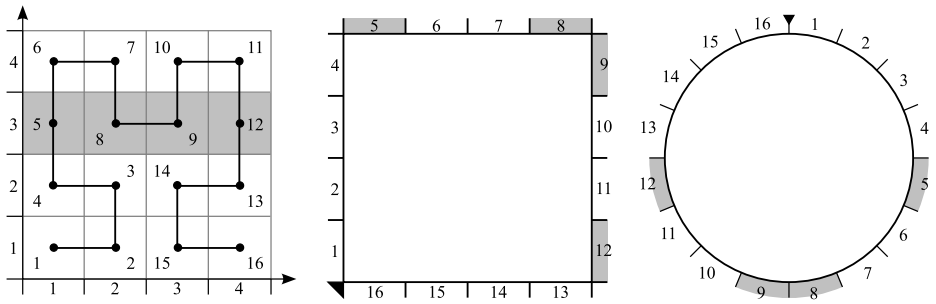



Abbildung 2.6: Zu sehen ist eine beispielhafte Bereichsanfrage, welche die Zonen 5, 8, 9 sowie 12 auf der Hilbert-Kurve ergibt, und wie der ursprüngliche Bereich auf die verteilte Hash-Tabelle abgebildet wird. Dabei zeigt sich die Lokalität des Clusters 8–9.

Die im letzten Abschnitt beschriebenen Cluster sind im Falle einer Bereichssuche wesentlich. Eine raumfüllende Kurve fasst zwar mehrere Dimensionen und damit auch Attribute zu einer Dimension, den Index, zusammen. Allerdings können bei einer Bereichssuche mehrere Indices berechnet werden. Ein einzelner Index ist ideal für eine Punktanfrage, bei der exakte Werte für jedes Attribute angegeben werden. Die Daten müssen folglich nur an dem Speicherort nachgesehen werden, auf den der Index verweist, anstatt alle Speicherorte nach den Daten zu durchsuchen. Eine solche Indexstruktur macht eine mehrdimensionale Suche sehr effizient. Mehrere Indices machen es erforderlich, die Daten an mehreren Orten ausfindig zu machen. Eine günstige Situation ergibt sich, wenn alle Indices nahe bei einander liegen, anstatt über den gesamten Wertebereich des Indexes verstreut zu sein. Denn dies reduziert die Entfernung der Speicherorte zueinander, wodurch eine Suche nicht deutlich teurer wird. Da eine raumfüllende Kurve alle Unterräume verbindet, werden benachbarte Schlüsselwerte von benachbarten Attributswerten aus abgebildet. Wie im obigen Beispiel zu sehen ist, trifft diese Eigenschaft für die umgekehrte Abbildung nicht zu: Benachbarte Attributswerte werden nicht immer auf benachbarte Indices abgebildet. Je weniger solcher Cluster eine Kurve bei einer Bereichssuche erzeugt, um so besser bewahrt sie die

Lokalität zwischen den ursprünglichen und den abgebildeten Werten. Diese Eigenschaft ist bei der Hilbert-Kurve am ausgeprägtesten (siehe Gotsman und Lindenbaum (1996); Mokbel u. a. (2003); Moon u. a. (2001)).

Mit der Verwendung von verteilten Hash-Tabellen, welchen das konsistente Hash-Verfahren zugrunde liegt (siehe das vorherige Kapitel 2.2 „Verteilte Hash-Tabellen“ ab Seite 16), zur Speicherung der Daten wird die Lokalität der Kurve ausgenutzt. Da einzelne Peers für Bereiche von Hash-Werten zuständig sind, könnten die Cluster einer Bereichssuche von nur einigen wenigen Peer ausgewertet werden. Neben der Indexstruktur ist dies ein weiterer Grund, warum die Anzahl der an einer Bereichssuche beteiligten Peers noch weiter sinken kann. Dies setzt jedoch voraus, dass die Anzahl der Zonen einer raumfüllenden Kurve größer ist als die Anzahl der Peers im Netz.

Neben der guten Lokalität bewahrenden Eigenschaft zeigt die Hilbert-Kurve über alle Dimensionen das gleiche Verhalten Mokbel u. a. (2003). Das bedeutet, es entsteht genau die selbe Anzahl von Clustern für einen Bereich, egal in welcher Dimension er definiert wird. Keine Dimension wird von der Kurve bevorzugt. Für die Indizierung von Daten ist diese Eigenschaft ebenfalls ideal, da es einerseits belanglos ist, welches Attribut welcher Dimension zugewiesen wird. Andererseits ist eine Bereichssuche in einer Dimension nicht kostengünstiger als in einer anderen Dimension, da das Verhalten der Kurve über alle Dimensionen gleich verteilt ist.

Allen guten Eigenschaften zum Trotz besitzt die Hilbert-Kurve auch eine nicht unerhebliche, schlechte Eigenschaft in Bezug auf die Anzahl der Dimensionen. Es wurde gezeigt, die durchschnittliche Anzahl der Cluster fängt ab der fünften Dimension an, exponentiell zu wachsen (siehe Moon u. a. (2001)). Während circa 2.500 Cluster für acht Dimensionen noch akzeptabel erscheinen, so äußert der Autor der Untersuchung selbst Zweifel, ob eine Benutzung der Hilbert-Kurve für Daten mit hoher Dimensionalität sinnvoll ist, da die durchschnittliche Anzahl der Cluster für zehn Dimensionen bereits 19.683 beträgt. Hier müssen weitere Recherchen angestrengt werden, ob andere raumfüllenden Kurven diesbezüglich ein besseres Verhalten aufweisen. Allerdings könnte dies zu einem Kompromiss führen, da die Hilbert-Kurve in Bezug auf Lokalität und Gleichbehandlung der Dimensionen besser ist.

# Kapitel 3

# Verwandte Arbeiten

Die bestehenden Ansätze für multidimensionale Bereichsanfragen in Peer-to-Peer-Netzen können in zwei Kategorien unterteilt werden. In der einen Kategorie werden spezielle logische Netzstrukturen aufgebaut, die von den Attributen abhängen, und in der anderen Kategorie werden verteilte Hash-Tabellen als Grundlage für Datenstrukturen verwendet. Letztere kann weiter unterteilt nach der Art, wie die Attribute in die Datenstrukturen mit einfließen. Zum einen werden individuelle Indexstrukturen für jedes einzelne Attribut aufgebaut und zum anderen werden Indexstrukturen über alle Attribute verwendet. Zudem gibt es den Ansazt dieser Diplomarbeit, die mehrere Indexstrukturen über Attributskombinationen nutzt. Die nachfolgenden Kapitel stellen Ansätze aus den jeweiligen Kategorien vor.

## 3.1 Spezialisierte Netzstrukturen

Ein Vertreter der spezialisierten Netzstrukturen ist Mercury (siehe Bharambe u. a. (2004)). In diesem System werden die Knoten zu logischen Ansammlungen namens Routing-Hub (*routing hub*) zusammengeschlossen. Für jedes Attribut im Anwendungsgebiet wird eine solche Ansammlung erstellt. Hierbei kann ein physischer Knoten in mehreren dieser logischen Hubs vertreten sein. Die Hubs sind in einem Ring organisiert und jeder Knoten ist für einen Bereich des Attributs zuständig. Daten werden gespeichert, indem sie an alle Hubs gesendet werden. Eine Anfrage hingegen wird nur mit einem Hub durchgeführt, dessen Attribut mit einem Attribut der Anfrage übereinstimmen muss.

Anstelle von mehreren Ringen wird in Datta u. a. (2005) eine Netzstruktur aufgebaut, die einem Trie entspricht, wobei die Funktionalität einer verteilten Hash-Tabelle geboten wird. Das heißt, es wird ein Speichern von und

ein Suchen nach Daten realisiert. Ein Trie wird in Datenbanken verwendet und ist eine Art Baum für Zeichenkettenpräfixe. Für jeden gemeinsamen Präfix gibt es einen Knoten im Baum. Dabei werden semantisch nahe Datenobjekte, die einen gemeinsamen Präfix aufweisen, nahe beieinander im Baum angehäuft.

Der Suchaufwand wird bei beiden Ansätzen als logarithmisch angegeben und ergibt sich aus der speziell abgestimmten Netzstruktur. Allerdings ist die Wartung solcher System aufgrund der komplexen Struktur sehr kostenintensiv. Da solche Kosten bei verteile Hash-Tabellen deutlich günstiger ausfallen, werden sie bevorzugt verwendet.

## 3.2 Verteilte Hash-Tabellen basierte Netze

Verteilte Hash-Tabellen werden einerseits aufgrund ihrer geringen Wartungskosten und andererseits wegen ihrer effizienten Schlüsselsuche als Grundlage für Datenstrukturen verwendet, die komplexere Anfragen erlauben, als die Suche nach nur einem Bezeichner. Bisher sind dem Autor zwei Ansätze für multidimensionale Bereichsanfragen bekannt. Zum einen gibt es Indexstrukturen, die für jedes einzelne Attribut erstellt werden. Bei einer Anfrage werden ihrer Attribute entsprechende Indexstrukturen verwendet und die Ergebnisse auf dem Urheberpeer aggregiert, womit die Kommunikationskosten bei der Abarbeitung unnötig hoch sind. Zum anderen werden Indexstrukturen über alle Attribute eingesetzt. Diese erzeugen ebenfalls einen hohen Kommunikationsaufwand für Anfragen, die nicht alle Attribute des Indexes enthalten. Daher wird ein dritter Ansatz vorgestellt, welcher in dieser Diplomarbeit erarbeitet wurde. Hierbei werden mehrere Indexstrukturen über Attributskombinationen verwendet.

### 3.2.1 Individueller Index

Der Präfix-Hash-Baum (*prefix hash tree*, siehe Ramabhadran u. a. (2004)) nutzt die verteilte Hash-Tabelle für einen Trie, welcher bereits in Kapitel 3.1 „Spezialisierte Netzstrukturen“ ab Seite 25 beschrieben wurde, wobei der Trie hier nicht die Netzstruktur bestimmt. Die Idee ist eine Hashfunktion auf die Präfixe anzuwenden und damit die Knoten zu bestimmen, die für den Präfix verantwortlich sind. Dies hat die Vorteile der Anhäufung semantisch ähnlicher Daten durch den Trie und einer schnelle Suche nach dem Knoten, welcher einen Präfix verwaltet durch die verteilte Hash-Tabelle. Bereichsanfragen werden durchgeführt, indem zuerst der Knoten mit der verteilten Hash-Tabelle bestimmt und anschließend eine Travesierung der Blätter

durchgeführt wird.

Mit MAAN (*multi-attribute addressable network*, mit mehreren Attributen adressierbares Netz, siehe Cai u. a. (2003)) wird ein anderer Ansatz verfolgt, bei dem ein Chord-Ring benutzt wird. Da die Hash-Funktion SHA-1 die Lokalität der Schlüsselwerte zerstört, wird für Zahlen stattdessen eine Hash-Funktion verwendet, die diese Lokalität bewahrt, womit die Zahlen geordnet auf den Chord-Ring abgebildet werden. Eine Suche nach einem Zahlenbereich, der durch eine untere und eine obere Grenze angegeben ist, beginnt bei der unteren Grenze und verläuft linear durch alle Knoten bis zur oberen Grenze. Eine Bereichssuche über mehrere Attribute wird durchgeführt, in dem die Bereichssuche den Chord-Ring anhand von einem Attribut durchläuft und auf den entsprechenden Knoten die Daten gesammelt werden, welche allen Attributsbereichen der Anfrage entsprechen.

Während beim Präfix-Hash-Baum mehrere Attribute jeweils nur einzeln gespeichert werden müssen, erhöht sich dieser Aufwand bei MAAN für die Bereichsanfrage, da alle Attributewerte der Daten für jedes Attribut notwendig sind, damit alle Attributbereich einer Anfrage abgearbeitet werden können, obwohl der Ring nur für ein Attribut durchlaufen wird. Wie zu sehen ist, entstehen auch hierbei Kosten, die vermieden werden können, wie die nachfolgenden Systeme zeigen.

### 3.2.2 Gesamt-Index

SCRAP (siehe Ganesan u. a. (2004)) bedient sich einer raumfüllenden Kurve, um mehrere Dimensionen auf eine abzubilden, und eines sogenanten Skip-Graphen. Bei letzterem handelt es sich um eine zirkuläre verkette Liste von Knoten, die auch Verweise über große Entfernungen enthält. Die Suche nach einem mehrdimensionalen Attribut erfolgt durch eine Abbildung mittels der raumfüllenden Kurve auf eine Dimension und der Suche im Skip-Graphen. Bei Bereichsanfragen werden alle eindimensionalen Werte bestimmt und ebenfalls im Skip-Graphen nach diesen gesucht.

Das Squid-System (siehe Schmidt und Parashar (2003)) hingegen verfolgt einen anderen Ansatz. Es baut auf dem Chord-Ring auf und bedient sich der Hilbert-Kurve für die Umrechnung mehrerer Dimensionen auf eine. Anstatt jedoch die eindimensionalen Werte komplett zu berechnen, wird der rekursive Aufbau der Hilbert-Kurve ausgenutzt und die Kurve bei der Abarbeitung einer Anfrage Schritt für Schritt verfeinert. Das heißt, es wird die Kurve der nächsten Annäherungsstufe berechnet.

Beide Ansätze verwenden für Anfragen eine Indexstruktur, die mit allen Attributen aufgebaut wurde und damit einen Gesamt-Index darstellt. Für Anfragen mit weniger Attributen stellt dies keinen optimalen Index dar,

denn es muss der gesamte Wertebereich für fehlende Attriubte angenommen werden. Daher wird der im nachfolgenden Kapitel beschriebene Ansatz verfolgt.

### 3.2.3 Indices für Attributskombinationen

Der in dieser Diplomarbeit erarbeitete Ansatz, verwendet mehrere Indexstrukturen aus Attributskombinationen. Hierbei handelt es sich um einen hybriden Ansatz, da mehrere individuelle Indexstrukturen verwendet werden, diese aber aus mehreren Attributen bestehen. Somit ist er eine Mischung aus den beiden vorher beschriebenen, auf verteilte Hash-Tabellen basierenden Ansätzen.

Über eine Heuristik wird eine gewisse Anzahl an Attributskombinationen ausgewählt, die als Indexstruktur verwendet werden. Zudem wird ein Algorithmus vorgestellt, welcher aus dieser Menge einen optimalen Index für eine Anfrage auswählt, mit dem die Anfrage abgearbeitet wird. Dies soll zu einer minimalen Anzahl an Peers führen, welche in die Bearbeitung der Anfrage involviert sind, und damit sowohl Kommunikationskosten sparen als auch die Leistungsfähigkeit erhöhen. Das Systemmodell und die Konzepte der Optimierung werden im nachfolgenden Kapitel präsentiert.

# Kapitel 4

# Konzepte zur Optimierung

Nachdem in Kapitel 2 „Grundlagen“ ab Seite 13 die Voraussetzung für das Verständnis dieser Arbeit gelegt und im vorherigen Kapitel andere Systeme vorgestellt wurden, werden das in dieser Diplomarbeit verwendete Systemmodell sowie die erarbeiteten Konzepte zur Optimierung multidimensionaler Bereichsanfragen beschrieben und erläutert. Zuerst wird ein allgemeiner Überblick gegeben, der alle Komponenten des Systems kurz beschreibt und der Orientierung dient. Anschließend werden die einzelnen Komponenten, insbesondere der in Kapitel 3.2.3 „Indices für Attributskombinationen“ ab Seite 28 vorgestellte Ansatz, sowie deren Funktionsweisen aufgezeigt. Hierbei wird aus der Perspektive einer Anwendung vorgegangen, die das System anwendet. Dies soll helfen, sowohl den Zusammenhang der einzelnen Komponenten als auch das gesamte Systemmodell an sich besser zu verstehen.

## 4.1 Übersicht

Es sind drei Teile aus denen sich das Systemmodell zusammensetzt. Oben auf sitzt das Anwendungsgebiet, welches sich das System zu eigen macht. Die für eine effiziente Suche verwendete Grundlage bilden verteilte Hash-Tabellen, wobei Chord als ein Vertreter solcher Peer-to-Peer-Netze ausgewählt wurde. Die dazwischenliegende Mittelschicht bildet der Lösungsansatz dieser Diplomarbeit, der es ermöglicht effiziente Bereichsanfragen über mehrere Attribute durchzuführen. Die Abbildung 4.1 auf Seite 30 verdeutlicht den Aufbau.

Das Anwendungsgebiet bestimmt die Attribute der Daten und somit auch der Indexstrukturen. Als Beispielanwendung wurde die Suche nach Ressourcen im Bereich des sogenannten Grid Computings verwendet. Dabei handelt es sich um von verschiedenen Betreibern verwaltete Rechner-Ressourcen,

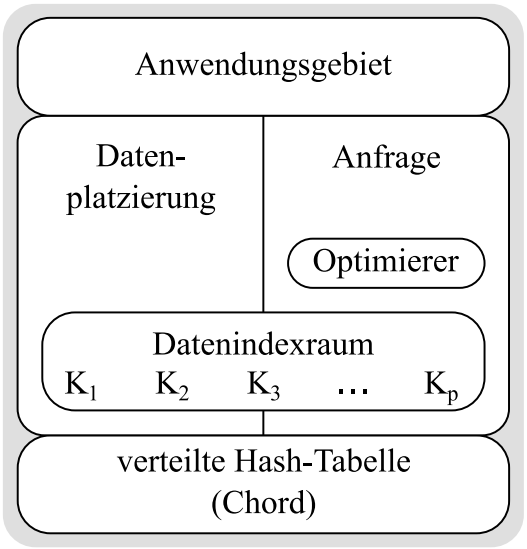



Abbildung 4.1: Das Systemmodell besteht aus drei Schichten. Diese sind die benutzende Anwendung (oben), der Ansatz dieser Diplomarbeit (Mitte) sowie die verwendete verteilte Hash-Tabelle (unten). Der Datenindexraum enthält die spezialisierten raumfüllenden Kurven (K), die von der Datenplatzierungs- und Anfragekomponente benutzt werden.

die verteilt und gegenseitig zur Verfügung gestellt werden (siehe Kesselman und Foster (1998)). Mit Hilfe des hier vorgestellten Systems kann die Suche nach solchen Ressourcen auf Basis des Peer-to-Peer-Netzes effizient durchgeführt werden. Die wesentliche Optimierung gegenüber anderen Peer-to-Peer-Systemen liegt in der Auswahl eines anfrage-spezifischen Indexes, der eine optimale Bearbeitung einer Suchanfrage garantiert.

Nachdem die Attribute des Anwendungsgebietes definiert wurden, gilt es die einzelnen Indexstrukturen zu bestimmen, damit die Hilbert-Kurven entsprechend eingesetzt werden können. Anschließend werden mit Hilfe der Datenplatzierungskomponente die Daten auf den Peers im Netz gespeichert. Eine Suche nach den Daten kann mit der Anfragekomponente durchgeführt werden. Letztere hat neben der Abarbeitung der Anfrage auch die Aufgabe einen optimalen Index auszuwählen, damit die Suche mit minimalen Kosten durchgeführt werden kann.

Das zugrunde liegende Peer-to-Peer-Netz dient einzig und allein der Suche nach den die Daten besitzenden Peers. Währende bei der Platzierung der Daten im Netz nur einzelne, punktuelle Bezeichner des Chord-Rings erfragt werden, sind es bei der Abarbeitung einer Bereichsanfrage durch die Anfragekomponente Bezeichnerbereiche. Der direkte Austausch der Daten

zwischen den Netzknoten wird nicht über das Chord-Protokoll abgewickelt. Da es eine reine Implementierung einer verteilten Hash-Tabelle darstellt, sind andere Aufgaben mit dem Protokoll nicht durchführbar. Diese muss das System selbst übernehmen. Dafür kann aufgrund dieser strikten Trennung neben dem Chord-Protokoll auch jede andere Umsetzung einer eindimensionalen verteilten Hash-Tabelle verwendet werden, die entweder wie in Chord implizit mit dem Erfragen eines Bezeichners einen Bereich zurückliefert oder diese Funktionalität explizit zur Verfügung stellt. Dies ist von Vorteil, sollte es in Zukunft noch effizientere oder für Bereichsanfragen spezialisierte Lösungen geben.

## 4.2 Daten- und Anfrage-Modell

Die Beschreibungsdaten der Ressourcen werden in Datenobjekten definiert. Diese wiederum werden im Peer-to-Peer-Netz dezentral gespeichert und enthalten zudem einen Verweis auf den tatsächlichen Ort der Ressource, was beispielsweise eine Internet-Adresse sein kann. Diese Indirektion oder indirekte Speicherung sollte soweit möglich bei anderen Arten von Daten ebenfalls angewandt werden. Denn würden die Daten selbst direkt auf den Peers gespeichert, so müssten diese anstelle der Beschreibungsdaten beim Ein- oder Austritt eines Peers übertragen werden, wenn sich das Netz reorganisiert. Handelte es sich dabei um große Datenmengen, führte dies zu einem hohen Kommunikationsaufkommen. Das Peer-to-Peer-Netz sollte nur dem Indizieren und effizienten Auffinden von Daten dienen. Sollen ebenfalls große Datenmengen im selben Netz gespeichert werden, so muss dieser Mehraufwand in Kauf genommen werden.

Die Tabelle 4.1 auf Seite 32 zeigt die Definition der im Anwendungsbeispiel verwendeten Attribute. Jedes Attribut hat einen Namen und einen Wertebereich, der durch ein Minimum und ein Maximum definiert wird. Die Einheit bestimmt die Größenordnung eines Attributs, wobei die Beschreibung die Bedeutung erklärt. Für ein Datenobjekt müssen alle Attribute definiert werden, was einer Menge von Name-Wert-Paaren entspricht, die als Tupel angegeben werden. Ein Beispiel einer Definition ist:

$$\begin{aligned}\text{Obj} = \; & \{ (\texttt{'CPU-Takt'}, \texttt{'2,0'}), (\texttt{'CPU-Last'}, \texttt{'20,7'}), \\ & (\texttt{'HD-Größe'}, \texttt{'200'}), (\texttt{'HD-Platz'}, \texttt{'142'}), \\ & (\texttt{'RAM-Größe'}, \texttt{'4,0'}), (\texttt{'RAM-Nutzung'}, \texttt{'38,0'}), \\ & (\texttt{'Bandbreite'}, \texttt{'5,8'}) \}.\end{aligned}$$

Sollte für ein Attribut kein Wert angegeben werden können, so muss ein Wert bestimmt werden, der „kein Wert“ definiert. Ansonsten müssten für

den nicht definierten Wert alle möglichen Werte aus dem Wertebereich des Attributs angenommen werden. Damit ist eine Indizierung nicht möglich, außer es würden für alle Werte eigene Datenobjekte erzeugt. Dies führte zu einem enormen Platzverbrauch bei der Speicherung im Peer-to-Peer-Netz und wird daher nicht unterstützt.

| **Name** | **Min.** | **Max.** | **Einheit** | **Beschreibung** |
|---|---|---|---|---|
| CPU-Takt | 0,1 | 4,0 | GHz | Taktfrequenz des Prozessors |
| CPU-Last | 0,0 | 100,0 | Prozent | Beanspruchung der maximalen CPU Leistungsfähigkeit |
| HD-Größe | 10 | 3000 | GB | Größe der Festplatte (*hard disk*, HD) |
| HD-Platz | 10 | 3000 | GB | frei verfügbarer Festplattenplatz |
| RAM-Größe | 0,5 | 8,0 | GB | Größe des Hauptspeichers (*random access memory*, RAM) |
| RAM-Nutzung | 0,0 | 100,0 | Prozent | Beanspruchung des Hauptspeichers |
| Bandbreite | 0,5 | 100,0 | Mbps | Bandbreite der Netzschnittstelle |

Tabelle 4.1: Definition einiger Attribute

| | | | |
|---|---|---|---|
| $=$ | gleich | $\neq$ | ungleich |
| $<$ | kleiner | $>$ | größer |
| $\leq$ | kleiner oder gleich | $\geq$ | größer oder gleich |

Tabelle 4.2: Operatoren einer Anfrage

Eine Suchanfrage (*query*) wird zusammengesetzt aus den Namen des Attributs, einem Operator und einem Wert: `Name` *Operator* `Wert`. Die Tabelle 4.2 auf Seite 32 listet die verwendeten Operatoren auf. Es können mehrerer solcher Terme mit einem logisch Und (`&`) verknüpft werden, wie das folgende Beispiel zeigt:

`CPU-Takt>3,0 & HD-Platz>1000 & RAM-Größe=4,0.`

Es wird zwischen zwei Anfragetypen unterschieden. Der eine Typ sind die Anfragen, welche das System wie oben beschrieben übergeben bekommt. Daraus

wird ein Anfragetyp mit normierter Syntax generiert, welcher den eigentlichen Anfrage innerhalb des System entspricht. Letzere wird bei der Suche nach den Daten verwendet und ist für die Anwendung transparent. Hierfür werden die Angaben übersetzt, die entweder einen Wert oder einen Bereich für alle angegebenen Attribute enthalten. Dabei müssen nicht alle Attribute vorkommen, da für fehlende Attribute alle möglichen Werte angenommen werden, was mit einem Stellvertreterzeichen (*wild card*) realisiert wird. Die Übersetzung ist notwendig, weil es nicht möglich ist, alle Operatoren direkt umzusetzen, und bietet der Anwendung gegenüber eine von den Implementierungsdetails abstrahierte Syntax. Außerdem kann der Anwendung später eine mächtigere Anfragesyntax zur Verfügung gestellt werden, die in einfache atomare Anfragen zerlegt wird, welche im Netz abgearbeitet werden können. Der Operator für die Ungleichheit kann beispielsweise nicht ohne weiteres im System benutzt werden. Stattdessen müssen zwei Anfragen erzeugt werden. Die eine Anfrage sucht nach kleineren und die andere nach größeren Werten. Eine vollständige Behandlung solcher komplexen Anfragen mitsamt der automatischen Generierung der für das Netz passenden Anfragen sprengt den Rahmen dieser Arbeit und wird hier deshalb nicht weiter betrachtet.

## 4.3 Datenindexraum

Der Datenindexraum beinhaltet alle Räume sowie die Abbildungen zwischen diesen, die für die Berechnung der Indices aus den Daten erforderlich sind. Hierzu zählen in erster Linie sowohl die Attributskombinationen, die einen Datenraum aufspannen, sowie die zugehörigen raumfüllenden Kurven, welche für die Berechnung der Indices verwendet werden. Außerdem wird noch die Abbildung der Indizes auf die Bezeichner des Chord-Rings hinzugenommen. Der Datenindexraum muss vom Administrator des Systems konfiguriert werden, nachdem die Attributsdaten des Datenmodells festgelegt wurden. Die Schwierigkeit hierin liegt in der Auswahl der Attributskombinationen, die einen optimalen und anfrage-spezifischen Index bilden sollten. Im nachfolgenden Unterkapitel 4.3.1 „Bildung von Attribut-Untermengen“ ab Seite 34 wird dies diskutiert.

Für gewöhnlich wird im Chord-Ring eine Hash-Funktion verwendet, damit die Bezeichner möglichst gleichmäßig über alle Peers verteilt werden. Dies soll sicherstellen, dass die Last, sowohl des Speichers als auch der Anfragebearbeitung, gleichmäßig über das gesamte Netz verteilt ist. Eine solche Hash-Funktion zerstört jedoch die Lokalität bewahrende Eigenschaft der Hilbert-Kurve, womit die Suche aufwendiger und teurer wird. Statt eine Hash-Funktion auf die Indices anzuwenden, werden die Indizes direkt als

Bezeichner im Chord-Ring verwendet, um den für sie verantwortlichen Peer auszumachen. Allerdings wurde gezeigt, dass die in einem Peer-to-Peer-Netz gespeicherten Daten nicht gleich häufig vorkommen (siehe Ripeanu (2001); Shu u. a. (2005); Klemm u. a. (2004); Gish u. a. (2007)). So gibt es wenige, aber dafür sehr populäre Daten, die sehr häufig vorhanden sind. Im Gegensatz dazu gibt es sehr viele, wenig populäre Daten, die deutlich weniger im Netz gespeichert sind. Eine ungleichmäßige Verteilung der Daten führt zu einem ungleichmäßig verteilten Index. Wenn beispielsweise ein Attributswert sehr populär ist und die anderen Attributswerte nur leicht variieren, könnten aufgrund der Lokalität der Hilbert-Kurve nur wenige Netzknoten für die Daten zuständig sein. Hieraus folgt, Peers könnten überlastet werden, wenn sie für Bezeichner zuständig sind, die von sehr populären Daten her resultieren. Folglich bedarf es eines expliziten Lastausgleichs (*load balancing*). Hinzu kommt die Dynamik der Daten im Netz selbst, da im Laufe der Zeit Daten entfernt und neu hinzugefügt werden. Damit ändert sich ebenfalls die Verteilung der Daten, weshalb ein Lastausgleich notwendig wird.

Das Thema Lastausgleich ist nicht Bestandteil dieser Arbeit und wird daher nicht näher untersucht. Jedoch können bei einer entsprechenden Abbildung der Indices auf den Chord-Ring Situationen auftreten, die Lastspitzen begünstigen oder hervorbringen. Aus diesem Grund wird auf diese Problematik in Kapitel 4.3.2 „Anwendung der raumfüllenden Kurven“ ab Seite 37 in Bezug auf die Abildung eingegangen.

### 4.3.1 Bildung von Attribut-Untermengen

In Kapitel 3.2.2 „Gesamt-Index“ ab Seite 27 wurden Peer-to-Peer-Netze vorgestellt, die über alle Attribute einen Index bilden. Diese Art der Indexbildung ist nicht optimal. Wird nur nach einigen wenigen Attributen gesucht, müssen für alle anderen Attribute alle Werte aus den jeweiligen Wertebereichen angenommen werden, was sowohl bei einer Punkt- als auch bei einer Bereichssuche zu einer großen Anzahl unnötig berechneter Indices führt. Je mehr Indices sich für eine solche Anfrage ergeben, desto mehr Speicherorte müssen kontaktiert werden. Die Berechnung unnötig vieler Indices kann vermieden werden, wenn anfrage-spezifische Indices aufgebaut werden. Die Vorteile liegen klar auf der Hand: Optimale Indices führen zu der minimal notwendigen Anzahl von Indices für die Bearbeitung von Anfragen, weshalb die Durchführung einer Suche kostengünstiger wird. Der Lösungsansatz dieser Diplomarbeit besteht darin, solche optimalen, anfrage-spezifischen Indexstrukturen aufzubauen. Hierfür werden Attributskombinationen gebildet, die als Datenindexraum dienen. Die Bildung von Attribut-Untermengen ist wie folgt definiert.

**Definition 1** *Gegeben sei eine Menge von Attributen* $A = \{a_1, a_2, a_3, \ldots, a_n\}$ *mit* $n \in \mathbf{N}$. *Es werden* $p \in \mathbf{N}$ *Attribut-Untermengen* $A_1, A_2, A_3, \ldots, A_p$ *gebildet, so dass* $\forall i, j \in [1, p]$, $i \neq j$ :

$$A_i \subseteq A, A_i \neq A_j, \mid A_i \mid > 1 \text{ und } \bigcup_{i=1}^{p} A_i = A.$$

Es werden folglich $p$ Attribut-Untermengen gebildet, die mindestens zwei Attribute enthalten müssen. Keine Untermenge darf einer anderen gleichen, was zu gleichen Indices führte. Zudem müssen alle Attribute in der Vereinigung der Untermengen vertreten sein, so dass alle Attribute indiziert werden.

Die Beschränkung auf $p$ Attributmengen liegt in der großen Anzahl von Kombinationen begründet. Bei $n$ Attributen gibt es $2^n - n - 1$ mögliche Untermengen. Die Herleitung der Berechnungsformel ist in Anhang A.1 „Anzahl der Attributskombinationen“ ab Seite 79 zu finden. Das Wachstum liegt in der Komplexitätsklasse $\mathcal{O}(2^n)$ und ist damit exponentiell. Würde die Anzahl nicht beschränkt werden, könnten bereits mit einer kleinen Anzahl von Attributen eine große Anzahl von Untermengen und damit Indexstrukturen entstehen. Jedoch bedeuten mehrere Indexstrukturen einen höheren Speicherbedarf im Netz, da jedes Datenobjekte mit jeder Indexstruktur indiziert wird. Das heißt, jedes Datenobjekt ist im Peer-to-Peer-Netz $p$ mal gespeichert. Die $p$ optimalen Kombinationen für die Indizierung auszuwählen ist ein komplexes Problem, denn es müssen viele Faktoren berücksichtigt werden. Hierzu gehören sowohl die Abbildung der Hilbert-Bezeichner auf den Bezeichnerring als auch die Zuordnung der Bezeichnerbereich an die Peers. In einem dynamischen Peer-to-Peer-Netz kann sich letzteres ändern, was bedacht werden muss. Es könnte sein, dass eine Indexstruktur, die nicht exakt mit den Attributen einer Anfrage übereinstimmt, aufgrund der Bezeichnerverteilung im Netz eine bessere Leistungsfähigkeit aufzeigt als eine komplett übereinstimmende, welche theoretisch als optimal gilt. Da die Häufigkeit der Anfragen in einem Peer-to-Peer-Netz nicht gleichverteilt ist (siehe Ripeanu (2001); Shu u. a. (2005); Klemm u. a. (2004); Gish u. a. (2007)), wurde eine Heuristik ausgewählt, die sich an der Popularität orientiert. Daher werden die $p-1$ häufigsten Anfragekombinationen ausgewählt. Die Bildung von Attribut-Untermengen nach relativer Häufigkeit von Anfragen ist wie folgt definiert.

**Definition 2** *Gegeben sei eine Menge von Anfragen* $Q = \{q_1, q_2, q_3, \ldots, q_{p-1}, q_p, \ldots, q_n\}$ *mit* $p, n \in \mathbf{N}$ *sowie eindeutigen Attributskombinationen und sei* $\mathcal{P}(q)$ *die (statistische) Wahrscheinlichkeit einer Attributskombination* $q$. $A_{q_i}$ *sei die Attribut-Untermenge von* $q_i$ *und es gilt:* $\mathcal{P}(q_1) \geq \mathcal{P}(q_2) \geq \mathcal{P}(q_3) \geq$

$\ldots \geq \mathcal{P}(q_{p-1}) \geq \mathcal{P}(q_p) \geq \ldots \geq \mathcal{P}(q_n)$. *Es werden* $p$ *Attribut-Untermengen* $A_1, A_2, A_3, \ldots, A_p$ *gebildet, so dass* $\forall i, j \in \mathbf{N},\ i \in [1, p-1]$ :

$$A_i = A_{q_i} \text{ und } A_p = \bigcup_{j=p}^{n} A_{q_j}.$$

Für die $p-1$ häufigsten Anfragen werden deren Attribute-Kombinationen direkt als Attribut-Untermenge für die Indexstrukturen übernommen. Mit den Attributskombinationen der restlichen $n-p$ Anfragen wird die Vereinigungsmenge gebildet, welche die letzte Attributskombination darstellt. Sollte die relative Häufigkeit mehrerer Anfragen gleich sein, so werden Anfragen zufällig ausgewählt, bis die Anzahl von $p-1$ erreicht ist.

Die Idee hinter dieser Auswahl ist, mit einem optimalen Index für eine bestimmte Anfrage keinen Mehraufwand in Bezug auf die Leistungsfähigkeit der Suche für sehr häufige Anfragen zu erzeugen, da sich bereits ein kleiner Mehraufwand aufgrund der großen Häufigkeit stark auswirkt. Bei Anfragen, die weniger oft durchgeführt werden, fällt ein Mehraufwand deutlich geringer ins Gewicht, weshalb hier kein optimaler Index eingesetzt wird. In der Evaluation in Kapitel 5.1 „Experiment 1 — raumfüllende Kurven“ ab Seite 60 wird gezeigt, eine nicht exakte Übereinstimmung der Attribute zwischen einer Indexstruktur und einer Anfrage führt zu einer erhöhten Anzahl von zu kontaktierenden Peers. Im schlimmsten Fall wird die letzte Attribut-Untermenge alle Attribute enthalten ($A_p = A$), jedoch ist der mit diesen Attributen verbundene Mehraufwand nicht schlechter als andere Ansätze, die nur einen einzigen Index über alle Attribute erstellen. Im Übrigen sei nochmals darauf hingewiesen, dass die Reihenfolge der Attribute im Datenindexraum hierbei nicht von Bedeutung ist, weil die Hilbert-Kurve alle Dimensionen gleich behandelt, wie in Kapitel 2.3 „Raumfüllende Kurven“ ab Seite 19 bereits erwähnt wurde.

Bevor spezialisierte Indices gebildet werden können, muss erst in Erfahrung gebracht werden, welche Suchanfragen in einem Netz gestellt werden. Aus diesen lassen sich die häufigsten Attributskombinationen bestimmen. Entweder es liegen bereits Statistiken über Suchanfrangen eines Netzes vor, welche direkt verwendet werden können. Oder es muss zuerst ein nicht optimiertes Netz aufgebaut werden, welches beispielsweise über alle Attribute einen Index erstellt, in dem dann die statistischen Daten gesammelt werden. Es sei darauf hingewiesen, Anfragen können sich genauso wie die Daten selbst über die Zeit ändern, weshalb ein bereits bestehendes Netz ständig die Attribute der Anfragen erfassen und auswerten muss, damit die Indexstrukturen bei Bedarf angepasst werden können.

Für zukünftige Arbeiten wird kurz auf die Problematik der Auswahl von Attributskombinationen eingegangen. Sollte die relative Häufigkeit aller Anfragen annähernd gleich sein, wird es schwer, die richtigen $p - 1$ Kombinationen auszuwählen. Anstatt wie weiter oben beschrieben, eine zufällige Auswahl zu treffen, könnten beispielsweise fast-optimale Indices gebildet werden, bei denen für maximal ein Attribut ein Stellvertreterzeichen verwendet werden darf. Der Mehraufwand, im Sinne berechneter Indices (siehe Kapitel 2.3 „Raumfüllende Kurven“ ab Seite 19), ist hierbei noch linear mit der Anzahl der Dimensionen ($2^k \cdot 2^{k \cdot d}$), wohingegen mehr als ein Attribut einen exponentiellen Mehraufwand bedeuten ($2^{a \cdot k} \cdot 2^{k \cdot d}$). Allerdings ist der Mehraufwand in beiden Fällen exponentiell zur Annäherungsstufe. Es bleibt auch zu klären, wie Anfragen zu handhaben sind, die untereinander Untermengen bilden: sowohl $A_i \subseteq A_k$ als auch $A_j \subseteq A_k$ oder sogar $A_i \subseteq A_j \subseteq A_k$. Hier gilt es ebenfalls eine optimale Indexstruktur zu finden, wobei diese nicht anfragespezifisch sein muss. Diese beiden Beispiele und weitere Sonderfälle bedürfen weiterer Untersuchungen.

### 4.3.2 Anwendung der raumfüllenden Kurven

Sind die Attributskombinationen festgelegt, werden drei Abbildungen durchgeführt. Zuerst wird der Raum, den die Attribute aufspannen, auf einen Raum für die raumfüllenden Kurven abgebildet, in dem jede Dimension den gleichen Wertebereich hat. Anschließend wird über die Berechnung der Hilbert-Kurve ein eindimensionaler Raum erzeugt. Zum Schluss wird die Kurve auf den Chord-Ring abgebildet. Die Abbildungen können mathematisch wie folgt dargestellt werden, wobei $\mathbf{N}$ die Menge der natürlichen Zahlen, $d$ die Anzahl der Dimensionen und $k$ die Anzahl der Annäherungsstufen sind:

1. $\mathbf{N}^d \mapsto [2^k]^d$
2. $[2^k]^d \mapsto [2^{k \cdot d}]$
3. $[2^{k \cdot d}] \mapsto [2^m]$

Die Abbildung 4.2 auf Seite 38 veranschaulicht diese Abbildungen. Alle drei Abbildungen werden für jede Attributskombination durchgeführt und daher existieren für jede von ihnen eine individualisierte raumfüllende Kurve. Nachfolgend werden die Abbildungen erläutert.

Der Wertebereich jedes Attributs kann individuell sein. Die Anwendung der Hilbert-Kurve setzt allerdings voraus, dass alle Dimensionen den gleichen Wertebereich aufweisen. Daher muss der mehrdimensionale Raum der Attribute skaliert werden, damit die Voraussetzung erfüllt ist. Hierfür muss der

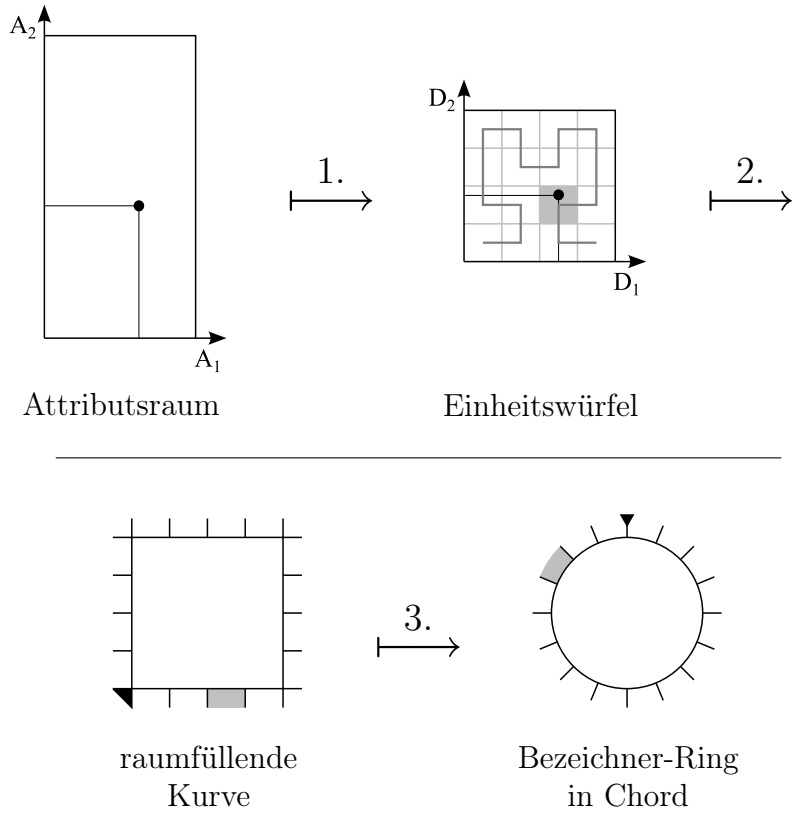



Abbildung 4.2: Es ist die Abbildungskette zwischen den einzelnen Räumen zu sehen. Der Punkt markiert ein Datenobjekt. Zuerst wird der Attributsraum auf den Einheitswürfel der raumfüllenden Kurve abgebildet (1.). Daraufhin wird der Hilbert-Bezeichner des Datenobjektes berechnet (2.). Dies entspricht einer Zone auf der Kurve, welche grau gefärbt ist. Abschließend wird die raumfüllende Kurve auf den Bezeichner-Ring des Chord-Protokolls abgebildet (3.).

Verwalter des Systems eine maximale Annäherungsstufe $k_{\max}$ festlegen, woraus sich der Wertebereich $[2^{k_{\max}}]$ jeder Dimensionen ergibt. Dabei sollte der größte Wertebereich aller Attribute einer Attributskombination als Richtwert dienen, insbesondere wenn es sich um diskrete Werte handelt. Damit wird die „Auflösung“ aller Attribute gewahrt, das heißt, feinste Abstufungen bleiben erhalten. Sind in der Attributkombination sowohl Attribute mit einem sehr kleinen als auch Attribute mit einem sehr großen Wertebereich enthalten, werden die kleineren Bereich unnötig weit gestreckt. Daher könnte es günstiger sein, Attribute mit ungefähr gleichgroßen Wertebereichen zu einer Attributkombination zusammenzufassen. Dies macht die im vorherigen Kapitel diskutierte Bildung von Attribut-Untermengen noch komplizierter und

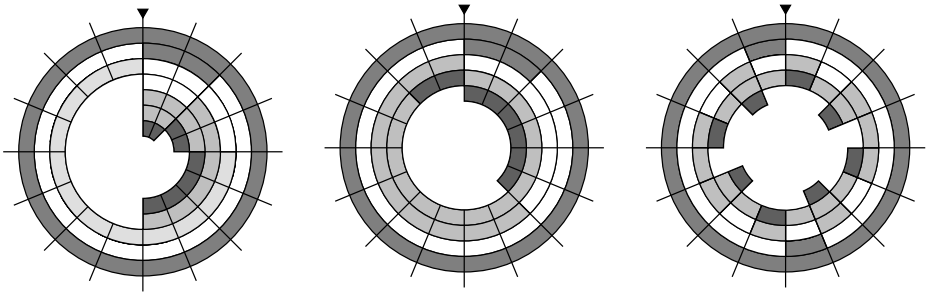
soll nur als Anreiz für weitere Überlegungen dienen. Zudem wirkt sich die erste Abbildung auch auf andere Bereiche des Systems aus. Wird einerseits ein Wertebereich unnötig weit gestreckt, so geht die Lokalität in der entsprechenden Dimension verloren. Wird dagegen ein Wertebereich stark gestaucht und gehen damit Abstufungen verloren, so werden unterschiedliche Werte verstärkt auf den gleichen Index abgebildet. Dies macht einen Lastausgleich schwerer, wie er eingangs in Kapitel 4.3 „Datenindexraum“ ab Seite 33 diskutiert wurde, da ein Index eine atomare Einheit darstellt. Die Aufspaltung eines Indexes ist mit einem Mehraufwand in der Verwaltung verbunden und erhöhte die Komplexität des gesamten Systemmodells.

Die zweite Abbildung entspricht der Berechnung der raumfüllenden Kurve, welche die Hilbert-Bezeichner und damit den Index repräsentiert. Sie wurde bereits im Kapitel 2.3 „Raumfüllende Kurven“ ab Seite 19 ausgiebig diskutiert, weshalb hier nicht näher darauf eingegangen wird.

Die letzte Abbildung der Hilbert-Kurve auf den Chord-Ring wird benötigt, um mehrere Indexstrukturen auf einem Chord-Ring zu verwalten. Sie ist in Sytemen mit nur einer Indexstruktur nicht notwending, da dort nur eine raumfüllende Kurve gleichzeitig in einem Chord-Ring verwendet wird und der Bezeichnerbereich des Rings an den Wertebereich der Kurve angepasst wird. Das ist im Falle mehrerer Indices nicht möglich, wenn es unterschiedlich große Bezeichnerbereich gibt. Eine Möglichkeit besteht darin, eine entsprechend hohe Annäherungsstufe der Kurven zu wählen, damit die Kurve sich dem Chord-Ring annähert. In dieser Diplomarbeit werden mehrere Indices verwendet und da nicht alle Kurven zwingend den gleichen Bereich aufweisen müssen, kann keine Anpassung des Bezeichnerbereichs durchgeführt werden. Die Wahl einer höheren Annäherungsstufe ist ebenfalls keine Option, da dies nur unnötig den Wertebereich „aufbläht“, obwohl kaum Daten für die zusätzlichen Indices zu erwarten sind. Daher werden hier Abbildungsstrategien erläutert die letztendlich auf die ausgewählte Abbildung hinführen.

Der Chord-Ring verwendet $2^m$ Bezeichner für die verteilte Hash-Tabelle, wobei in der ursprünglichen Version $m = 160$ ist. Aufgrund der Zweierpotenz wird hier auch von $m$ Bits gesprochen, da mindestens $m$ Bits für die Repräsentation des Bezeichners verwendet werden müssen. Es wurde bereits in Kapitel 4.3 „Datenindexraum“ ab Seite 33 erläutert, dass eine Hash-Funktion für die Abbildung nicht in Frage kommt, da sie die Lokalität zerstört. Die einfachste Abbildung besteht darin, die $2^{k \cdot d}$ Zonen der raumfüllenden Kurve direkt auf den Bezeichnerraum des Rings abzubilden. Sollte der Bereich überschritten werden, wird modulo $2^m$ gerechnet. Allerdings kann dies zu einer Überlast am Anfang des Bereichs führen. Das heißt, die Peers, welche für die Bezeichner am Anfang des Bereichs zuständig sind, können eine große Anzahl an Datenobjekten erhalten, da jede Kurve beim ersten Wert beginnt.

Dorthin wird nicht nur der Anfang einer jeden Kurve abgebildet, sondern zu große Bereiche beginnen am Anfang mit der Überlappung auf dem Ring. Die Abbildung 4.3 auf Seite 40 veranschaulicht diese einfache Abbildung. Selbst ein Lastausgleich kann die Situation nicht verbessern, da die Kurven alle dieselben anfänglichen Indexwerte besitzen und ein Index die atomare Einheit im System ist.



$2^{k \cdot d} \bmod 2^m$ $\quad$ $(2^{k \cdot d} \bmod 2^m) + c$ $\quad$ $(s \cdot 2^{k \cdot d}) + c$

Abbildung 4.3: Die Schaubilder zeigen die drei vorgestellten Abbildungsmöglichkeiten der $2^{k \cdot d}$ Hilbert-Bezeichner auf die $2^m$ Bezeichner des Chord-Rings.

Mit einem Versatz (*offset*) der einzelnen Kurven um einen konstanten Faktor $c$ können die Bereiche der Kurven besser abgestimmt werden, so dass die Überlappungen ausgeglichener sind: $(2^{k \cdot d} \bmod 2^m) + c$. Dies wird in der Abbildung 4.3 auf Seite 40 dargestellt. Diese Berechnung ist gültig für Bereiche, die sowohl kleiner als auch größer als der Ringbereich sind. Allerdings kommt es bei dieser Lösung darauf an, wieviele Kurven erstellt werden und wie groß deren Bereiche sind. Sollten die Kurven nicht den ganzen Bereich des Ringes füllen, bleibt ein konzentrierter Bezeichnerbereich ohne jegliche Indices. Ist die Summe der Bereiche jedoch größer als der Ringbereich, so gibt es einen konzentrierten Bereich in dem die Indices überlappen. Zudem hängt bei beiden Vorschlägen die Belastung der einzelnen Peers davon ab, für welche Bezeichner sie zuständig sind und wie die Verteilung der Daten ist. Ein Lastausgleich muss folglich auch die Abbildung auf den Chord-Ring beachten.

Neben der direkten Abbildung mitsamt einer Verschiebung besteht die Alternative der Skalierung. Dabei wird der Bereich um einen Faktor gestreckt, wenn er auf den Ring abgebildet wird: $s \cdot 2^{k \cdot d}$. Bereiche, die zu groß sind, werden auf den Ring gestaucht, wobei die gleiche Anzahl von mehrfach belegten Indices entsteht, wie bei der direkten Abbildung. Dies wird im Anhang A.2 auf Seite 79 bewiesen. Bereiche, die kleiner als der Ringbereich sind, werden dagegen gleichmäßig auf den Ring verteilt. Der Vorteil ist, die Bereiche, auf

die entweder kein Index oder eine große Zahl von Indices abgebildet wird, verteilt sich über den gesamten Bezeichnerring. Ein Lastausgleich sollte von der bereits vorhandenen Verteilung profitieren. Jedoch kommt es hier ebenfalls auf die Verteilung der Daten an, inwiefern ein Lastausgleich notwendig ist. Der Skalierungsfaktor lässt sich wie folgt berechnen:

$$\begin{aligned} s \cdot 2^{k \cdot d} &= 2^m \\ s &= \tfrac{2^m}{2^{k \cdot d}} \\ s &= 2^{m-k \cdot d}. \end{aligned}$$

Da es sich hier ausschließlich um Zweierpotenzen handelt, kann die Skalierung effizient als Bitschiebeoperation (*bit shifting*) um $m - k \cdot d$ nach links realisiert werden. Negative Werte bedeuten entsprechend eine Verschiebung nach rechts.

Bei der Skalierung können ebenfalls Überlappungen der einzelnen Indices auftreten, wenn mehrere Abbildungen den gleichen Skalierungsfaktor verwenden. Dabei werden alle Indices der betroffenen raumfüllenden Kurve auf den gleichen Bezeichner im Chord-Ring abgebildet. Daher sollte hier ebenfalls ein Versatz verwendet werden, der die Kurven gegeneinander verschiebt: $s \cdot 2^{k \cdot d} + c$. Die Verschiebung macht nur Sinn, wenn der Bereich der Kurve kleiner als der Bereich des Ringes ist ($2^{k \cdot d} < 2^m$), da ansonsten alle Bezeichner des Rings in Anspruch genommen werden. In Abbildung 4.3 auf Seite 40 wird dieser Ansatz visualisiert.

Die Skalierungslösung kann den Vorteil der verteilten Hash-Tabelle zu Nichte machen, bei dem ein Peer für mehrere Bezeichner verantwortlich ist. Kleine Wertebereiche der raumfüllenden Kurve werden durch die Skalierung über den gesamten Bezeichnerbereich des Chord-Rings verteilt. Solange die Anzahl der Peers klein genug ist, erhält jeder Peer mit hoher Wahrscheinlichkeit mindestens einen Indexwert zugewiesen. Je mehr solcher Indexwerte einem Peer zugewiesen sind, desto größer ist der Bereich, den er bei einer Bereichsanfage abarbeiten kann. Sind genau so viele Peers vorhanden, wie es Schlüsselwerte gibt, erhält jeder Peer mit hoher Wahrscheinlichkeit nur einen einzigen Wert. Folglich erhöht sich bei einer Bereichsanfrage die Anzahl der Knoten, die kontaktiert werden müssen, wobei die Lokalität noch bewahrt bleibt, da der Nachfolger eines Peers für den nächsten Indexwert verantwortlich zeichnet. Ist die Anzahl der Peers dagegen deutlich größer als die Anzahl der Schlüsselwerte, gibt es Peers, die keinen solchen Wert erhalten. Damit ist sogar die über die Hilbert-Kurve erhaltene Lokalität nicht mehr gegeben, weil sich die Entfernung der Peers vergrößert, die einen Bezeichner der Kurve haben. Dies führt zu einem höheren Kommunikationsaufkommen und damit zu einer teureren Suche. Wird von einer maximalen Annäherungsstufe von 8 ausgegangen, sind es $2^8 = 256$ diskrete Werte für jede Dimension. Für

zwei Dimensionen sind es „nur“ $2^{8 \cdot 2} = 65.536$ Zonen und damit Indexwerte. Bei drei Dimensionen sind es bereits $2^{8 \cdot 3} = 16.777.216$ Werte. An den Zahlenbeispielen ist zu sehen, dass bei nur 8 Annäherungsstufen bereits drei Dimensionen ausreichen, um eine ausreichende Anzahl an Zonen für Peer-to-Peer-Netze mit einigen Millionen Teilnehmern zu erhalten. Für Netze mit nur einigen Tausend Teilnehmern sind bereits zwei Dimensionen ausreichend. Dies zeigt ebenfalls, die Abbildungsstrategie kann abhängig von der Bildung der Attribut-Untermengen gewählt werden. Würden minimal nur drei Dimensionen zugelassen, fiele diese Eigenschaft nicht so sehr ins Gewicht, sofern die maximale Annäherungsstufe hoch genug ist.

## 4.4 Datenplatzierungskomponente

Die Aufgabe der Datenplatzierungskomponente ist recht simpel. Insbesondere da sie auf den Datenindexraum aus dem vorherigen Kapitel zurückgreift, der die gesamte Abbildung von den Daten bis hin zum Bezeichner im Chord-Ring übernimmt. Sie ist dafür zuständig, Datenobjekte im Peer-to-Peer-Netz sowohl zu speichern als auch zu löschen. Da beide Vorgänge den gleichen Ablauf haben, wird hier nur der Speichervorgang beschrieben. Hierfür erhält die Komponente von der Anwendung ein Datenobjekt, das wie in Kapitel 4.2 „Daten- und Anfrage-Modell“ ab Seite 31 definiert sein muss. Aus dem Datenobjekt werden mittels aller Indexstrukturen die zugehörigen Indices berechnet. Anschließend wird die Funktionalität des Chord-Protokolls genutzt, den Peer zu finden, der für einen Index zuständig ist. Dieser wird daraufhin direkt kontaktiert und das Datenobjekt wird ihm übermittelt. Damit ist das Datenobjekt im Netz platziert. Der Vorgang wiederholt sich für jeden Index, bis alle Datenobjektkopien im Netz gespeichert wurden.

Es ist klar ersichtlich, dass der Aufwand mit der Anzahl der Indexstrukturen steigt, weil für jede Indexstruktur eine Kopie des Datenobjekts gespeichert werden muss. Dies garantiert jedoch eine effiziente Suche für Anfragen, die exakt die selben Attribute aufweisen, für die die Indexstruktur erstellt wurde. Dabei wird zugunsten einer effizienteren Suche ein höherer Speicherplatzverbrauch in Kauf genommen (*time-space-trade-off*). Im Vergleich zu einem einzigen Index erhöht sich der Speicherplatzbedarf des in dieser Diplomarbeit verwendeten Ansatzes um den Faktor $p$. Wobei $p$ die Anzahl der verschiedenen Attributskombinationen und damit der Indexstrukturen ist. Auch wenn der Festplattenspeicherplatz heutzutage billig ist, sollten aus diesem Grund keine umfangreichen Daten im Datenobjekt gespeichert werden. Daher empfiehlt sich bei größeren Daten eine Indirektion, wie sie bereits in Kapitel 4.2 „Daten- und Anfrage-Modell“ ab Seite 31 besprochen wurde. Der

vermeintliche Nachteil des erhöhten Speicheraufkommens kann jedoch für die Ausfallsicherheit genutzt werden. Anstatt, wie in solchen Peer-to-Peer-Systemen üblich, mehr Redundanz in das Netz zu bringen, indem explizit Replikate der Daten gespeichert werden, sind in diesem System die Replikate bereits implizit vorhanden. Allerdings mangelt es an einem Algorithmus, der die mehrfach vorhandenen Datenobjekte dazu nutzt, eine Ausfallsicherheit und Fehlertoleranz des Systems aufzubauen. Diese Aufgabe sollte mit dem Lastausgleich gekoppelt werden, damit letzterer die Zuverlässigkeit des Systems nicht schwächt, und bleibt zukünftigen Arbeiten überlassen.

## 4.5 Anfragenkomponente

Die Suche nach Datenobjekten stellt sich im Vergleich zur Platzierung aufwendiger dar. Neben der Übersetzung von komplexen Anfragen, wie sie in Kapitel 4.2 „Daten- und Anfrage-Modell“ ab Seite 32 beschrieben wird, gibt es zwei weitere Herausforderungen. Zum einen gilt es, diejenige Indexstruktur auszuwählen, welche die Anfrage am besten abbildet. Nur eine optimale Indexstruktur führt zu den günstigsten Kosten bei der Durchführung der Suche. Zum anderen sollte die Abarbeitung der Suchanfrage das Netz nicht mit Nachrichten überschwemmen, sondern kontrolliert vonstatten gehen.

Die Anfragenkomponente wählt die optimale Indexstruktur anhand bestimmter Kriterien aus und leitet die Suchanfrage an die Peers zur Abarbeitung weiter, die für die berechneten Indices zuständig sind. Diese werden mithilfe der verteilten Hash-Tabelle ermittelt. Zuerst wird eine grundlegende und einfache Weiterleitung vorgestellt. Anschließend wird auf Optimierungen eingegangen, die einerseits die maximale Anzahl der parallelen Nachrichten im Netz und andererseits den maximalen Rechenaufwand eines einzelnen Peers für eine Anfrage garantieren.

### 4.5.1 Auswahl einer Indexstruktur

Eine wesentliche Optimierung dieser Diplomarbeit besteht darin, für Suchanfragen eine Indexstruktur zu verwenden, welche die gleiche Attributskombination wie die Anfrage selbst aufweist. Dies stellt sicher, dass eine minimale Anzahl von Indices während der Abarbeitung der Anfrage verwendet wird und minimiert damit die Anzahl der Peers, die an der Abarbeitung der Anfrage beteiligt sind. Da die Anzahl der Attributskombination exponentiell wächst, wird nur eine bestimmte Anzahl solcher Kombinationen verwendet, um die Indexstrukturen aufzubauen. Dies wurde in Kapitel 4.3.1 „Bildung von Attribut-Untermengen“ ab Seite 35 bereits diskutiert. Das heißt, der optima-

le Index für eine bestimmte Anfrage könnte nicht in der Menge der im System verwendeten Indexstrukturen vorhanden sein. Daher ist ein Algorithmus notwendig, der eine optimale Indexstruktur aus den vorhandenen auswählt. Hier sollte klar sein, „optimal“ bedeutet, es wird die am besten passende Indexstruktur gewählt, welche nicht der exakten Attributskombination entsprechen muss. Es sei zudem darauf hingewiesen, der hier vorgestellte Ansatz ergibt im schlimmsten Fall eine Attributskombination, die einer Indexstruktur über alle Attribute entspricht. Damit wird die Leistungsfähigkeit in Bezug auf die Anfragebearbeitung im Netz nicht schlechter als die von bereits vorhandenen Systemen.

Generell beeinflussen sich die Auswahl der optimalen Indexstruktur und die Bildung von Attribut-Untermengen. Für erstere ungünstige Attributskombinationen können einerseits mit letzterer vermieden werden. Andererseits kann die Auswahl an die Bildung angepasst werden. Da die Bildung von Attribut-Untermengen weiterer Untersuchungen bedarf (siehe Kapitel 4.3.1 „Bildung von Attribut-Untermengen“ ab Seite 36), ist nicht auszuschließen, dass der nachfolgend beschriebene Algorithmus angepasst werden muss, wenn sich die Bildung der Attribut-Untermengen ändert, auch wenn er mit Bedacht entworfen wurde. Dies trifft ebenfalls zu, wenn sich die Abbildung der Indexwerte auf die Bezeichner der verteilten Hash-Tabelle ändert, da der Algorithmus auf dem Ansatz der Skalierung beruht (siehe Kapitel 4.3.2 „Anwendung der raumfüllenden Kurven“ ab Seite 40).

Bevor der Auswahlalgorithmus vorgestellt wird, bedarf es der Definition einer Metrik namens Zonenverhältnis. Diese dient als Hilfsmittel zur Bewertung, wie gut eine Indexstruktur zu einer Anfrage passt. Bei der Auswahl geht es in erster Linie darum, einen Index zu wählen, der zu einer geringen Anzahl von Peers führt, welche die Anfrage abarbeiten. Jede Indexstruktur definiert eine raumfüllende Kurve mit einer bestimmten Anzahl an Dimensionen $d$ sowie der maximalen Annäherungsstufe $k_{\max}$. Die Gesamtzahl aller Zonen und damit aller Indexwerte ist $2^{k_{\max} \cdot d}$, wie im Kapitel 2.3 „Raumfüllende Kurven“ ab Seite 21 nachzulesen ist. Sei $N$ die Anzahl der Peers im Netz und $z_{\max}$ die Anzahl aller Zonen in einer raumfüllenden Kurve, dann ist die Anzahl der Zonen, die ein Peer verwaltet, mit hoher Wahrscheinlichkeit

$$\frac{z_{\max}}{N} = \frac{2^{k_{\max} \cdot d}}{N}.$$

Die Division $\frac{z_{\max}}{N}$ setzt eine Gleichverteilung der Zonen einer raumfüllenden Kurve über alle Knoten voraus, was mit der Skalierung der Indexbereiche gegeben ist. Sei nun $z_q$ die Anzahl der Zonen für eine raumfüllende Kurve, die sich aus einer Anfrage $q$ ergibt. Die Anzahl der an der Bearbeitung

teilnehmenden Knoten lässt sich wie folgt abschätzen:

$$\frac{z_q}{\frac{z_{\max}}{N}} = z_q \cdot \frac{N}{z_{\max}} = \frac{z_q}{z_{\max}} \cdot N = \frac{z_q}{2^{k_{\max} \cdot d}} \cdot N.$$

Hier ist allerdings zu beachten, die Rechnung geht davon aus, dass alle Zonen eines Peers zur Anfrage gehören, was einer optimistischen Schätzung einer minimalen Anzahl von Peers entspricht. Diese muss nicht zutreffen, denn die Anfrage wird einerseits selten einen kontinuierlichen Indexbereich ergeben, sondern in einer Vielzahl von Clustern resultieren, welche über mehrere Peers verstreut sind. Andererseits ist die Zuständigkeit der Peers für die Zonen zufälliger Natur, weshalb nicht davon ausgegangen werden kann, dass ein Cluster von nur einem Peer bearbeitet wird. Denn ein Cluster kann sich über die Bereiche zweier Knoten erstrecken, auch wenn er weniger Zonen enthält als einem Peer mit hoher Wahrscheinlichkeit zugeteilt sind. Die Anzahl der Knoten im Peer-to-Peer-Netz ist für alle raumfüllenden Kurven gleich und spielt beim Vergleich zweier Kurven keine Rolle, wie folgende Beispielrechnung zeigt:

$$\begin{array}{lcl} \frac{z_{1,\mathrm{q}}}{z_{1,\max}} \cdot N & \stackrel{?}{=} & \frac{z_{2,\mathrm{q}}}{z_{2,\max}} \cdot N \quad \mid \cdot \frac{1}{N} \\ \frac{z_{1,\mathrm{q}}}{z_{1,\max}} & \stackrel{?}{=} & \frac{z_{2,\mathrm{q}}}{z_{2,\max}}. \end{array}$$

Nach der Erläuterung aller Parameter kann nun die verwendete Metrik definiert werden.

**Definition 3** *Gegeben sei eine raumfüllende Kurve mitsamt der Anzahl ihrer Dimensionen d sowie ihrer maximalen Annäherungsstufe $k_{\max}$, wobei gilt: $d, k_{\max} \in \mathbf{N}, d \geq 2$. Zudem sei eine Anfrage q gegeben, die in $z_q$ Zonen der raumfüllenden Kurve resultiert und es gilt: $1 \leq z_q \leq 2^{k_{\max} \cdot d}$. Das Zonenverhältnis wird definiert als*

$$Z = \frac{z_q}{2^{k_{\max} \cdot d}} \text{ , mit } Z \in \left[ \frac{1}{2^{k_{\max} \cdot d}}, 1 \right].$$

Das Zonenverhältnis bestimmt somit den Anteil an Zonen, in die eine Anfrage resultiert, im Verhältnis zur Anzahl aller Zonen der entsprechenden raumfüllenden Kurve. Es gibt mit einer optimistischen Schätzung an, wie viele Peers an der Abarbeitung einer Anfrage beteiligt sind, wenn eine bestimmte Indexstruktur verwendet wird. Der Bereich des Zonenverhältnisses geht von annähernd Null bis Eins und das heißt, die Anfrage wird von mindestens einem sowie höchstens $2^{k_{\max} \cdot d}$ Peers bearbeitet. Folglich bedeutet ein kleines Zonenverhältnis eine kostengünstige Abarbeitung der Anfrage. Es sei darauf hingewiesen, dass nicht mehr Peers beteiligt sein können, als eine raumfüllende Kurve Zonen hat.

Die Anzahl der Zonen und damit der Indexwerte für eine Bereichsanfrage kann bestimmt werden, ohne die Hilbert-Bezeichner der Zonen tatsächlich berechnen zu müssen. Diese Berechnung weist sogar den Vorteil auf, eine Indexstruktur auszuwählen, deren Attribute die meisten Übereinstimmungen mit den Attributen der Anfrage aufweist, ohne weitere Untersuchungen der Attribute durchführen zu müssen.

**Definition 4** *Die Berechnung des Zonenverhältnisses einer raumfüllenden Kurve für eine Anfrage sei wie folgt. Sei d die Anzahl der Dimensionen und $k_{max}$ die maximale Annäherungsstufe einer raumfüllenden Kurve mit $d, k_{max} \in \mathbf{N}$. Ferner sei $R = 2^{k_{\max}}$ die Anzahl aller diskreten Werte je Dimension sowie $r_i$ die Anzahl der diskreten Werte der i-ten Dimension bezüglich der Anfrage. Zudem bezeichnet $A_i$ das Attribut der i-ten Dimension und $A_q$ die Menge der Attribute der Anfrage q. Es gilt $\forall i \in \mathbf{N}, d \geq 2$:*

$$Z = \prod_{i=1}^{d} \frac{v_i}{R} \ , \ v_i = \begin{cases} r_i, & \textit{wenn } A_i \in A_q \\ R, & \textit{sonst} \end{cases} .$$

Stimmt das Attribut einer Dimension mit einem Attribut der Anfrage überein, so wird die Anzahl der diskreten Werte bestimmt, die der Bereich der Anfrage in dieser Dimension erzeugt. Falls es keine Übereinstimmung gibt, so muss der gesamte Wertebereich angenommen werden. Das Zonenverhältnis ist die Multiplikation der jeweiligen Verhältnisse aller Dimensionen, weil alle Dimensionen rechtwinklig auf einander stehen. Wie leicht zu sehen ist, entspricht das Produkt dem Zonenverhältnis aus Definition 3 auf Seite 45.

Interessant sind hier zwei Aspekte. Erstens werden Attribute in der Indexstruktur, die keine Übereinstimmung finden, nicht berücksichtigt, da $\frac{R}{R} = 1$ ist. Zweitens ist jenes Zonenverhältnis das kleinste, welches die meisten übereinstimmenden Attribute hat. Denn rein mathematisch ist $\frac{v}{R} < \frac{R}{R}$ für eine Anzahl diskreter Werte $v < R$. Beide Aspekte zeigen, das Zonenverhältnis ist nur eine notwendige aber keine hinreichende Bedingung. Für die Wahl der optimalen Indexstruktur ist das kleinste Zonenverhältnis notwendig, da dies garantiert, die kleinste Anzahl von Knoten an der Bearbeitung der Suchanfrage zu beteiligen. Allerdings ist es nicht ausreichend genug, weil nicht übereinstimmende Attribute nicht gewertet werden. Folglich kann bei gleichem Zonenverhältnis eine unterschiedliche Anzahl von nicht übereinstimmenden Attributen vorliegen. Je mehr nicht übereinstimmende Attribute vorhanden sind, desto mehr Zonen müssen bei der Abarbeitung berechnet werden. Daher ist eine hinreichende Bedingung nötig, die die Anzahl der Dimensionen berücksichtigt. Hingegen bedeuten weniger Dimensionen nicht zugleich eine bessere Indexstruktur, wenn die Anzahl der übereinstimmenden Attribute

ebenfalls kleiner ist, was ein größeres Zonenverhältnis ergibt. Deshalb werden beide, die notwendige und die hinreichende, Bedingung verwendet: kleinstes Zonenverhältnis und geringste Anzahl von Dimensionen. Während erstere die Kommunikationskosten senkt, verringert letztere den Rechenaufwand.

Der heuristische Algorithmus für die Auswahl einer Indexstruktur bedient sich sowohl der Metrik Zonenverhältnis als auch der hinreichenden Bedingung der geringste Anzahl der Dimensionen in einer priorisierten Reihenfolge. Es werden Bedingungen aufgestellt, die gelten müssen, damit eine Indexstruktur als optimaler Index für eine Anfrage ausgewählt wird. Sofern nur eine Struktur die Bedingung erfüllt, ist das Auswahlverfahren beendet. Andernfalls wird mit der nächsten Bedingung und der Index-Menge der vorherigen Bedingung fortgefahren, bis letztendlich die letzte Bedingung in Kraft tritt. Der Algorithmus arbeitet folgende Bedingungen nacheinander ab:

1. Wähle die Indexstruktur, deren Attribute mit den Attributen der Anfrage exakt übereinstimmen.
2. Wähle die Indexstruktur mit dem kleinsten Zonenverhältnis aus.
3. Wähle die Indexstruktur mit der geringsten Anzahl an Dimensionen aus.
4. Wähle die Indexstruktur zufällig aus.

Es sei angemerkt, die erste Bedingung ist nicht notwendig, da sie implizit über die Bedingung zwei und drei gegeben ist. Allerdings erspart das Überprüfen ersterer einige Berechnungen und macht das Auswahlverfahren verständlicher. Ferner fließt hier keinerlei Beachtung der von jeder raumfüllenden Kurve zu erzeugenden Cluster für eine Suchanfrage in die Auswahl mit ein. Das hat folgende Gründe. Zum einen müsste eine Analyse der Cluster vorhanden sein, bevor diese berechnet sind. Ein guter Hinweis für die Auswahl eines Indexes ist zum Beispiel die Anzahl der Cluster. Diese hängt allerdings davon ab, welche Bereiche in der Anfrage angegeben wurde. Zum anderen ist kein Wissen über die Verteilung der Zonen auf die Peers vorhanden, da es keine globale Sicht auf das Netz gibt. Dies macht eine genau Vorhersage über die Abarbeitung einer Anfrage sehr aufwendig, weshalb hier nur eine heuristische Auswahl getroffen wird. Bessere Verfahren bleiben zukünftigen Arbeit überlassen.

### 4.5.2 Grundlegende Anfrage-Weiterleitung

Ist die optimale Indexstruktur ausgewählt, werden alle Indexwerte für eine Anfrage berechnet. Diese entsprechen den Bezeichnern im Chord-Ring, da

eine Indexstruktur auch die Abbildung auf den Bezeichner-Ring durchführt. Handelt es sich dabei nur um einen Indexwert, wird der entsprechende Peer mithilfe des Chord-Protokolls ermittelt und die Anfrage zur weiteren Verarbeitung an ihn weitergesandt. Im Falle mehrerer Indexwerte, ist es sehr aufwendig für jeden einzelnen Schlüsselwert den Peer nachzusehen, zumal es sich dabei um eine sehr große Anzahl handeln kann. Da ein Peer in einer verteilten Hash-Tabelle für mehrere Bezeichner zuständig zeichnet, ist es für direkt benachbarte Indexwerte sehr wahrscheinlich, dass sie vom gleichen Peer verwaltet werden. Daher können Indexwerte zu Cluster zusammengefasst werden, welche in Kapitel 2.3 „Raumfüllende Kurven“ ab Seite 22 vorgestellt wurden. Hierbei handelt es sich um Indexwerte, die sich um höchstens einen Wert unterscheiden. Die grundlegende Weiterleitung einer Anfrage besteht nun darin, anstelle jedes einzelnen Indexwertes einen Cluster pro Nachricht zu versenden. Hierfür werden die Cluster in aufsteigender Reihenfolge sortiert. Für den ersten Schlüsselwert eines jeden Clusters wird der entsprechende Peer unter zur Hilfenahme der verteilten Hash-Tabelle ermittelt und diesem der gesamte Cluster mitsamt der ursprünglichen Anfrage übermittelt.

Bei Erhalt eines solchen Anfrage-Clusters verfährt ein Peer wie folgt. Da er ohne weitere Maßnahmen nur seinen eigenen Bezeichner kennt, entnimmt er dem Cluster alle Schlüsselwerte die kleiner oder gleich seines Bezeichners sind. Für diese arbeitet er die Anfrage ab und sendet eine Antwort an den Urheber. Den verbleibenden Cluster sendet der Peer wiederum an den Verantwortlichen des ersten Bezeichners. Die Weiterleitung erfolgt so lange, bis alle Indexwerte aus dem ursprünglichen Cluster entfernt wurden. In Abbildung 4.4 auf Seite 49 ist ein Beispiel einer solchen Weiterleitung gegeben.

In erster Linie ist die grundlegende Weiterleitung eine Art Fluten. Wird für eine Anfrage eine sehr große Anzahl von Clustern berechnet, werden all diese Cluster zeitgleich von einem Knoten aus versendet. Die Anzahl der ausgehenden Nachrichten hängt von dem Bereich der Anfrage sowie der verwendeten Indexstruktur und damit der Hilbert-Kurve ab. Ist die Anzahl der Attribute des Indexes größer als die Anzahl der Attribute der Anfrage, so steigt die Anzahl der berechneten Indexwerte deutlich an. Dies kann einerseits eine sehr große Anzahl an Clustern ergeben, die das Peer-to-Peer-Netz fluten. Andererseits könnten die Bereiche der Anfrage derart im Raum der Hilbert-Kurve liegen, dass weniger aber dafür sehr große Cluster entstehen. Sehr große Cluster führen zu einer sehr langen Weiterleitungskette, womit es länger dauert, bis die Anfrage für alle Bezeichner abgearbeitet wurde.

Die Gesamtzahl an Hilbert-Bezeichnern einer raumfüllenden Kurve mit beispielsweise 3 Dimensionen und einer maximalen Annäherungsstufe von 8, liegt bei $2^{3 \cdot 8} = 2^{24} = 16.777.216$. Eine Bereichsanfrage kann damit bereits zu einer Anzahl von mehreren Hundert bis zu Tausend von Clustern führen.

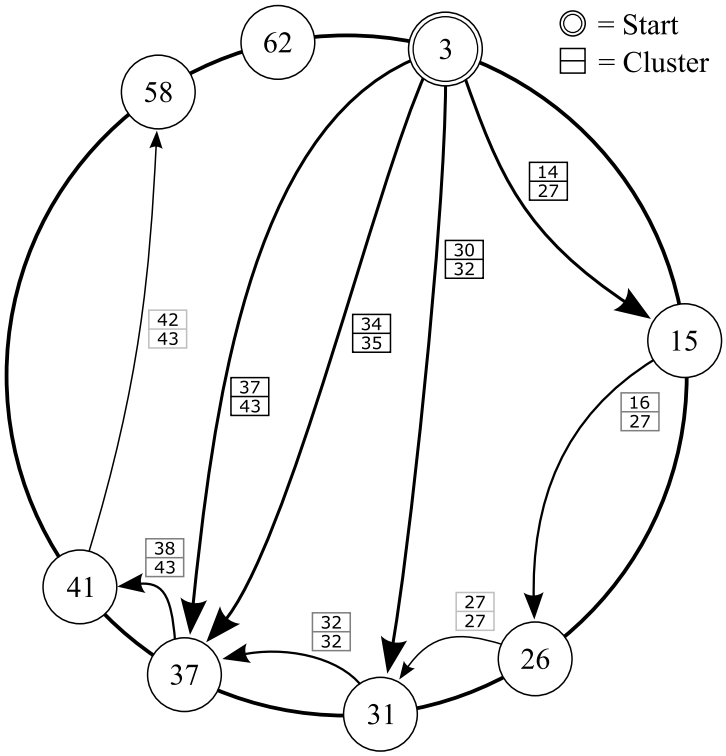



Abbildung 4.4: Es ist eine grundlegende Weiterleitung einer Anfrage zu sehen, welche im Peer mit dem Bezeichner 3 ihren Ursprung hat. Die Antwortnachrichten wurden zu Gunsten der Übersicht weggelassen.

Würden mehrere solcher Anfragen gleichzeitig im gesamten Netz durchgeführt, wäre das Nachrichtenaufkommen sehr hoch. Hinzu kommt, die Kosten für den Kommunikationsauf sowie -abbau, der Kommunikationsaufwand, zum Versenden sehr vieler kleiner Cluster fallen mehr ins Gewicht, als beim Versand weniger großer Nachrichten. Daher ist es wünschenswert, die Anzahl der zu versendeten Nachrichten bei einer Anfrage zu kontrollieren und zwar unabhängig von der Anzahl der Cluster. Eine solche Nachrichtenbeschränkung wird im nachfolgenden Kapitel vorgestellt.

### 4.5.3 Optimierung 1 — Nachrichtenbeschränkung

Die grundlegende Weiterleitung hat den entscheidenden Nachteil, von der Anzahl der Cluster abzuhängen. Da die Cluster parallel versendet werden, mag die Bearbeitungszeit der Anfrage im Netz sehr kurz sein. Doch bei einer großen Anzahl von Anfragen, die in einer großen Anzahl von Clustern resultieren, kann es zu einer Überflutung des Netzes kommen. Aus diesem Grund

wird eine Optimierung vorgestellt, die die Anzahl der parallelen Nachrichten für eine Anfrage kontrollierbar macht.

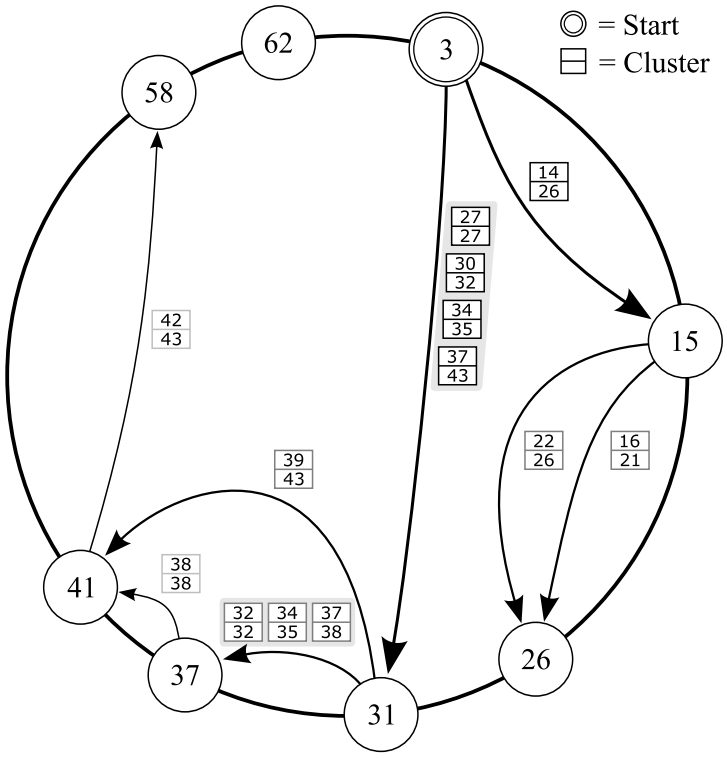



Abbildung 4.5: Es ist die erste Optimierung, die Nachrichtenbeschränkung, beim Weiterleiten einer Anfrage zu sehen, welche im Peer mit dem Bezeichner 3 ihren Ursprung hat. Die Antwortnachrichten wurden zu Gunsten der Übersicht weggelassen. Der Ausgangsgrad $f$ und die Baumtiefe $l$ sind beide 2. Wie zu sehen ist, teilt der Peer mit dem Bezeichner 41 den Cluster nicht mehr auf, weil hier bereits die Baumtiefe von 2 erreicht ist.

Ein erster Ansatz besteht darin, den Ausgangsgrad (*fan-out*, $f$) eines Knotens, welcher der Ursprung einer Anfrage ist, zu beschränken. Hierfür wird der Parameter $f$ definiert, der dessen Ausgangsgrad festlegt. Das heißt, es werden maximal $f$ Nachrichten für eine Anfrage von dem Knoten aus gesendet. Dazu werden die Cluster in aufsteigender Reihenfolge sortiert und auf $f$ Behältnisse verteilt. Jedes Behältnis wird anschließend mit der grundlegenden Weiterleitung aus dem vorherigen Kapitel versandt. Zwar beschränkt dieser Ansatz die Anzahl der parallelen Nachrichten, jedoch erhöht er auch die Anzahl der Cluster pro Nachricht, was zu einer sehr langen Weiterleitungskette auch mit nur kleinen Clustern führen kann. Daher scheint es günstig zu sein, $f$ groß zu wählen. Allerdings versendete erneut nur ein Knoten eine

große Anzahl an Nachrichten, wenn eine hohe Parallelität erwünscht ist, und damit fiele der Kommunikationsaufwand wieder mehr ins Gewicht.

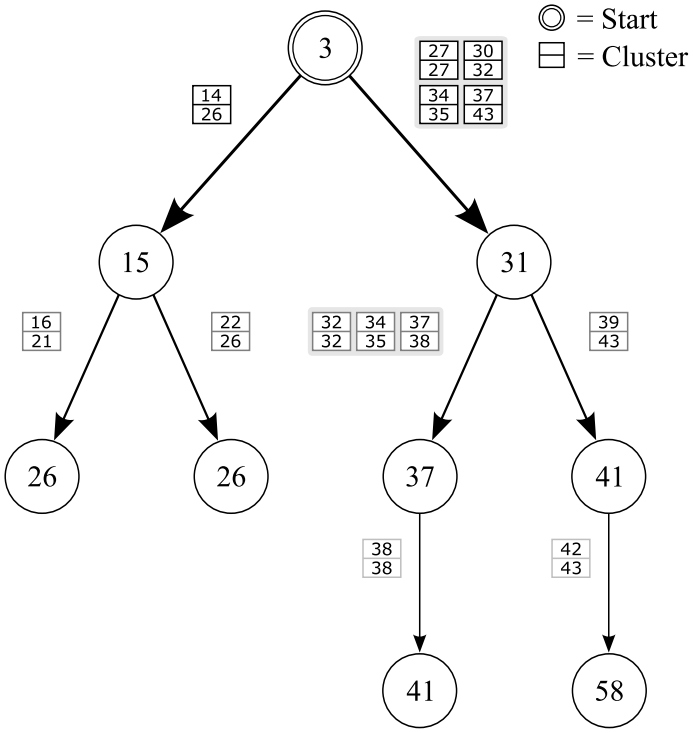



Abbildung 4.6: Es ist die Baumansicht der ersten Optimierung beim Weiterleiten einer Anfrage zu sehen, welche im Peer mit dem Bezeichner 3 ihren Ursprung hat. Die Antwortnachrichten wurden zu Gunsten der Übersicht weggelassen. Der Ausgangsgrad $f$ und die Baumtiefe $l$ sind beide 2. Wie zu sehen ist, teilt der Peer mit dem Bezeichner 41 den Cluster nicht mehr auf, weil hier bereits die Baumtiefe von 2 erreicht ist.

Die Beschränkung auf wenige Nachrichten eines Peers einerseits und eine hohe Gesamtzahl an gleichzeitigen Nachrichten andererseits erfordern einen weiteren Parameter namens Baumtiefe (*depth level*, $l$). Die Baumtiefe legt fest, wieviele an einer Anfrage beteiligte Peers die verbliebenen Cluster mit $f$ Nachrichten weiterleiten. Hierzu wird am Ursprungskonten die Baumtiefe festgelegt und beim Empfang einer Nachricht um eins heruntergezählt. Solange $l$ größer als Null ist, werden die verbliebenen Cluster mittels $f$ Nachrichten weitergeleitet. Wird $l$ dagegen Null, so kommt wieder die grundlegende Weiterleitung zum Einsatz. Die Bezeichnung Baumtiefe ist der in der Informatik bekannten Datenstruktur Baum und dessen Höhe entliehen, da die Weiterleitung der Nachrichten die Eigenschaften eines solchen Baumes bis

zur angegebenen Baumtiefe aufweist. In Abbildung 4.5 auf Seite 50 ist das Beispiel aus dem vorherigen Kapitel zu sehen, allerdings wird die Nachrichtenbeschränkgung angewandt. Der dazugehörige Baum ist in Abbildung 4.6 auf Seite 51 dargestellt.

Die Eigenschaften dieses Ansatzes sind sowohl ein geringerer Ausgangsgrad bei einem einzelnen Peer, als auch eine Kontrolle der maximalen Anzahl der parallelen Nachrichten für eine Anfrage im Netz. Während die Anzahl des Ausgangsgrades klar ersichtlich ist, kann die maximale Anzahl der parallelen Nachrichten leicht berechnet werden. Sie ist $f^l$. Die Herleitung der Berechnungsformel ist in Kapitel A.3 ab Seite 80 zu finden. Das Wachstum dieser Formel ist exponentiell in Bezug auf die Baumtiefe. Das heißt, mit jeder Weiterleitung steigt die Anzahl der Nachrichten einer Anfrage stark an. Je nach Wahl der Parameter, wird die maximale Anzahl schnell oder langsam erreicht. Wird beispielsweise $f$ sehr groß gewählt, bedarf es nur weniger Weiterleitungen. Ist hingegen $f$ sehr klein, ist die gewünschte Nachrichtenanzahl erst nach einer großen Anzahl von Weiterleitungen erreicht. Zu Gunsten einer schnellen Anfragebearbeitung kann ein hoher Ausgangsgrad festgelegt werden, wobei das Nachrichtenaufkommen im Netzwerk steigt. Wird hingegen das Nachrichtenaufkommen mittels des Ausgangsgrades reduziert, erhöht sich die Weiterleitungskette und damit die Suchdauer. Die Abbildung 4.7 auf Seite 52 veranschaulicht die Parameterwahl.

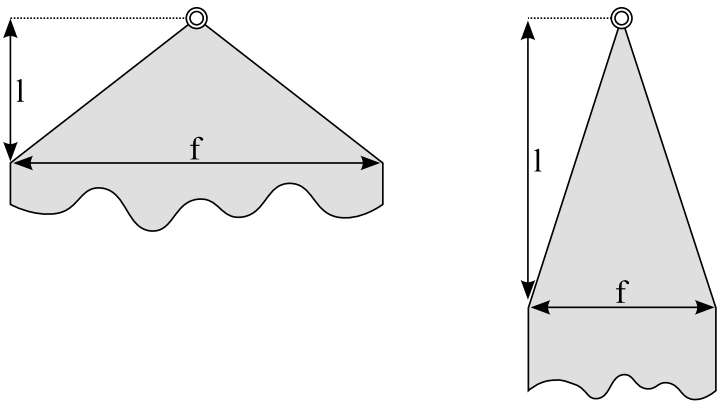



Abbildung 4.7: Die Darstellung zeigt den Einfluss der Parameter Ausgangsgrad $f$ sowie Baumtiefe $l$ in Bezug auf die Anzahl von Weiterleitungen, bis eine gewünschte maximale Anzahl von Nachrichten erreicht ist. Entweder wird $f$ groß gewählt, womit kleine $l$ ausreichen (links). Oder $f$ wird klein gewählt, was große $l$ erforderlich macht (rechts).

Es sei darauf hingewiesen, die Aufteilung aller Cluster in $f$ Behältnisse erfolgt anhand der Anzahl aller Zonen, da für eine optimale Aufteilung

mehr Aufwand betrieben werden müsste. Optimal wäre eine Aufteilung nach Bezeichner, für die die Peers im Netz verantwortlich sind. Da es an einer globalen Ansicht mangelt, können die Indexwerte nicht direkt den Knoten zugewiesen werden, ohne für jeden einzelnen Cluster den verantwortlichen Peer im Chord-Ring zu erfragen. Zudem hängen die Zonen ebenfalls von den Bereichen der Anfrage ab, was eine allgemein gültige Lösung erschwert, zumal das Peer-to-Peer-Netz sich dynamisch verhält und sich damit die Verantwortlichkeiten für Bezeichner ändern.

Bisher wurde nicht auf die Berechnungszeit eingegangen, die ein Peer aufwenden muss, bis alle Indexwerte für eine Anfrage berechnet werden. Da dies in der Praxis jedoch von Nachteil sein kann, wird die Berechnung über mehrere Peers verteilt. Hierauf wird im nachfolgenden Kapitel eingegangen.

### 4.5.4 Optimierung 2 — verteilte Berechnung

Die Berechnung aller Indexwerte einer Anfrage kann viel Zeit in Anspruch nehmen, da ihre Zeitkomplexität aufgrund der Hilbert-Kurve in $\mathcal{O}(2^{k \cdot d})$ liegt. Die Ursache liegt einerseits in der großen Anzahl der Schlüsselwerte selbst, wenn Bereiche der Anfrage großzügig gewählt werden, und andererseits in der beschränkten Rechenkapazität eines einzelnen Peers. Daher wird eine weitere Optimierung des Systems vorgestellt, bei der die Berechnung auf alle Knoten im Peer-to-Peer-Netz verteilt wird.

Die Verteilung der Berechnung sollte nur dann durchgeführt werden, wenn ein Peer mit ihr überfordert ist. Das heißt, der Rechner des Anwenders sollte innerhalb einer akzeptablen Zeit die Anfrage zur Bearbeitung weiterleiten. Hierfür muss ein Maß eingeführt werden, ab welchem Aufwand eine verteilte Berechnung notwendig ist, weil in der Praxis unterschiedliche Rechnersysteme zum Einsatz kommen und die Berechnungszeit individuell ist. Die Berechnung eines Indexwertes ist hierbei die atomare Berechnungseinheit, weshalb eine Größe angegeben wird, die bestimmt, wie viele Werte von einem einzelnen Peer für eine Anfrage berechnet werden. Es hat sich allerdings gezeigt, die Berechnung eines einzelnen Schlüsselwertes liegt für Prozessoren mit einer Taktung im Gigahertz-Bereich um die ein Hundert Millisekunden, weshalb die Angabe nicht direkt in Indexwerten, sondern mit dem Exponenten einer Zweierpotenz namens $B$ erfolgt: $2^B$. Dies korreliert mit der Berechnungsformel für die Gesamtzahl aller Zonen in der Hilbert-Kurve, welche $2^{k_{\max} \cdot d}$ ist. Denn anhand der Anzahl der Dimensionen und einer Anfrage kann bestimmt werden, wie viele Annäherungsstufen auf einem Peer berechnet werden. Das System unterstützt bisher die Berechnung der nächst höheren Annäherungsstufe für alle Cluster einer Nachricht auf einmal. Mit Mehraufwand kann die Annäherung auch für einzelne Zonen realisiert werden. Hierfür bedarf

es jedoch näherer Untersuchungen, ob der Nutzen die Kosten aufwiegt. Ist bei einer fortschreitenden Verfeinerung der Cluster die maximale Annäherungsstufe $k_{max}$ erreicht, muss die Berechnung nicht verteilt werden. Es wird garantiert, dass die Anzahl der berechneten Zonen nicht größer ist als $2^B$. Allerdings wird in jedem Fall die nächste Annäherungsstufe berechnet, um eine unendliche Weiterleitung ohne jegliche Berechnung zu vermeiden. Eine verteilte Berechnung ist mit der Hilbert-Kurve ohne weiteres möglich, da Zonen einer höheren Annäherungsstufe rekursiv erzeugt werden, indem bestehende Zonen geteilt werden. Daher muss kein weiterer Aufwand betrieben werden, der die Berechnung parallelisierbar macht.

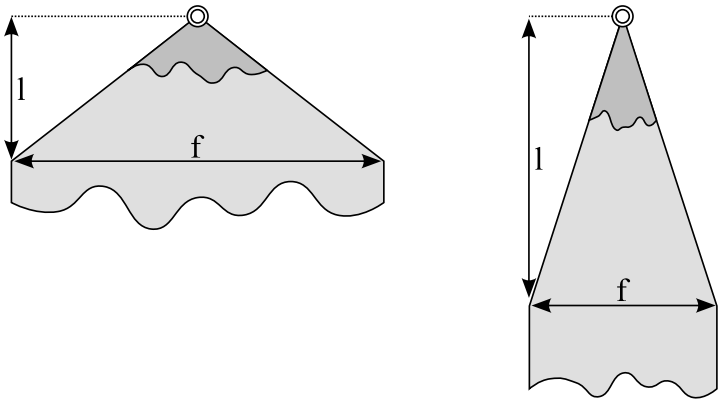



Abbildung 4.8: Die verteilte Berechnung erhöht die Anzahl der Weiterleitungen und damit die Baumtiefe. Sie beginnt am Ursprungsknoten einer Anfrage (dunkel gefärbte Baumspitze). Die Parameter Ausgangsgrad $f$ und Baumtiefe $l$ beeinflussen die Parallelität der Berechnung.

Ist eine Verteilung notwendig, da eine Bereichsanfrage die Berechnung einer Anzahl von Zonen erfordert, die größer als $2^B$ ist, so wird die Annäherungsstufe $k_q$ verwendet für die gilt: $\text{zonen}(q, 2^{k_q \cdot d}) \leq 2^B, k_q \geq 1$. Die Funktion zonen() berechnet hierbei die Anzahl der Zonen für eine Anfrage (*query*, $q$). Die Weiterleitung der Berechnung hält sich an die Nachrichtenbeschränkung und nutzt die Parameter $f$ und $l$. Es sei angemerkt, die Berechnung muss auf mehrere Peers gleichzeitig verteilt werden, damit von der Parallelität profitiert werden kann. Je mehr Peers gleichzeitig an der Berechnung teilnehmen, desto schneller wird diese fertiggestellt. Dies muss bei der Wahl von $f$ und $l$ berücksichtigt werden. Die Weiterleitung der Nachrichten erfolgt wie in der ersten Optimierung des vorherigen Kapitels, allerdings mit zwei Ausnahmen. Erstens muss zusätzlich zu der Anfrage und den bereits berechneten Clustern die Annäherungsstufe übermittelt werden, damit die nachfolgenden Berechnungen korrekt durchgeführt werden können, da

der Verlauf der Hilbert-Kurve sich mit der Annäherungsstufe ändert, wie in Abbildung 2.5 auf Seite 21 zu sehen ist. Zweitens werden die Nachrichten nicht an die Peers weitergeleitet, welche für die erste Zone eines jeden Clusters verantwortlich sind, sondern es wird zufällig ein Bezeichner aus dem Chord-Ring ausgewählt. Letzteres dient der Verteilung der Berechnung auf alle Peers, unabhängig von den gesuchten Daten. Damit wird vermieden, dass Peers mit Berechnungen überhäuft werden, deren Bezeichnerbereich populäre Daten beinhaltet. Ein verteiltes Berechenn findet auch im Squid-System Schmidt und Parashar (2003) statt, welches in Kapitel 3.2.2 „Gesamt-Index“ ab Seite 27 vorgestellt wurde. Jedoch werden dort die Anfragen an den Peer weitergeleitet, der für die Bezeichner einer Anfrage zuständig ist, weshalb Peers mit populären Daten nicht nur mit häufigen Anfragen, sondern auch mit deren verteilten Berechnung belastet werden. Die zufällige Auswahl des Bezeichners dagegen verteilt alle Berechnungen gleichmäßig auf die Peers, da jeder mit hoher Wahrscheinlichkeit einen gleichgroßen Bezeichnerbereich verwaltet. Hierfür sollte ein sehr guter Pseudozufallszahlengenerator verwendet werden, der auf kryptographischen Verfahren wie SHA-1 (siehe U.S. Dept. Commerce/NIST und National Technical Information Service (1995)) beruht, da diese eine sehr zufällige Verteilung aufweisen.

Es sollte klar sein, die verteilte Berechnung erhöht die Weiterleitungskette und damit die Zeit, bis alle Peers die Anfrage abgearbeitet haben. In Abbildung 4.8 auf Seite 54 wird dies skizziert. Im Vergleich zu einer sehr langen Berechnung aller Indexwerte auf einem einzelnen Peer ist die Verzögerung durch die verteilte Berechnung jedoch vernachlässigbar.

# Kapitel 5

# Systemevaluation

Das in den vorherigen Kapiteln beschriebene Systemmodell, die erarbeiteten Heuristiken, wie das Zonenverhältnis (siehe Definition 4 ab Seite 46) und der Auswahlalgorithmus für Indexstrukturen (siehe Definition auf Seite 47), sowie die beiden Optimierungen, Nachrichtenbeschränkung (siehe Kapitel 4.5.3 ab Seite 49) und verteiltes Berechnen (siehe Kapitel 4.5.4 ab Seite 53), wurden in einer Simulation bewertet, auf die in diesem Kapitel näher eingegangen wird.

Die Simulation wurde mit dem Simulator PeerSim (siehe PeerSim (2006)) in der Version 1.0.3 durchgeführt, der explizit für Peer-to-Peer-Netze entworfen wurde und in Java programmiert ist. Der Simulator bildet nur die Grundlage für die Kommunikation zwischen den Netzknoten. Eine bestehende Implementierung des Chord-Protokolls ist zwar vorhanden, allerdings mangelt es einerseits an einer Dokumentation und andererseits wurden bei einer Durchsicht kleine Fehler gefunden. Zudem bringt sie eine Funktionalität mit, die für die durchgeführten Tests nicht notwendig war. Daher wurde das Chord-Protokoll selbständig entwickelt. Für die raumfüllenden Kurven wurde nach einer Implementierung gesucht, um Entwicklungszeit einzusparen. Hierbei wurde der Quelltext von Lawder (2000) verwendet, welcher den Algorithmus aus Butz (1971) in der Programmiersprache C umsetzt. Neben der Übersetzung in die Programmiersprache Java musste ein kritischer Fehler aufgespürt und korrigiert werden, welcher verhinderte, dass zwischen einem Hilbert-Bezeichner und den Dimensionen nicht beliebig hin und her gerechnet werden konnte. Die Funktionen waren nicht invers zu einander. Zudem fehlte die Berechnung von mehreren Hilbert-Bezeichner für Bereich in den Dimensionen, welche ebenfalls selbständig ergänzt wurde. Alles zusammen hat viel Zeit bei der Implementierung in Anspruch genommen und es entstanden insgesamt 353 Kilobyte Quelltext, wobei 4.420 Zeilen Programmtext und 5.439 Zeilen Dokumentation, in Form von JavaDoc, geschrieben wurden.

Im Prototypen für die Simulation ist der Ansatz der Skalierung (siehe Kapitel 4.3.2 „Anwendung der raumfüllenden Kurven“ ab Seite 40) bei der Abbildung der Hilbert-Bezeichner auf den Bezeichnerring realisiert worden, da es noch an einer Lösung für den Lastausgleich mangelt und die Skalierung die Zonen aller raumfüllenden Kurven über den gesamt Bezeichnerbereich des Chord-Rings verteilt. Dies kommt einem Lastausgleich in Bezug auf die Zonenverteilung gleich, womit das gesamte Netz beansprucht wird.

| | **Name** | **Min.** | **Max.** | **Einheit** | **Beschreibung** |
|---|---|---|---|---|---|
| $\mathbf{A_1}$ | CPU-Takt | 0,1 | 4,0 | GHz | Taktfrequenz des Prozessors |
| $\mathbf{A_2}$ | CPU-Last | 0,0 | 100,0 | Prozent | Beanspruchung der maximalen CPU Leistungsfähigkeit |
| $\mathbf{A_3}$ | HD-Platz | 10 | 3000 | GB | frei verfügbarer Festplattenplatz |
| $\mathbf{A_4}$ | RAM-Größe | 0,5 | 8,0 | GB | Größe des Hauptspeichers (*random access memory*, RAM) |

| | $\mathbf{A_1}$ | $\mathbf{A_2}$ | $\mathbf{A_3}$ | $\mathbf{A_4}$ | **d** | $\mathbf{k_{max}}$ | **B** |
|---|---|---|---|---|---|---|---|
| $\mathbf{K_1}$ | √ | | √ | | 2 | 8 | 16 |
| $\mathbf{K_2}$ | √ | √ | | √ | 3 | 8 | 16 |
| $\mathbf{K_3}$ | √ | √ | √ | √ | 4 | 8 | 16 |

Tabelle 5.1: Es werden die in der Simulation verwendeten Attribute ($A$) und raumfüllenden Kurven ($K$) aufgelistet.

Die Attribute und die darauf aufbauenden raumfüllenden Kurven der Simulation sind in der Tabelle 5.1 auf Seite 58 zusammengefasst. Sie sind in allen Experimenten gültig. Die drei Kurven für die Indexstrukturen haben eine unterschiedlich große Anzahl an Dimensionen und es gelten in Bezug auf diese $K_1 < K_2 < K_3$ und in Bezug auf die Attribute $K_1 \subset K_3$ sowie $K_1 \subset K_3$. Die Parameter sind die Anzahl der Dimensionen $d$, die maximale Annäherungsstufe $k_{\max}$ sowie der Exponent $B$ für die maximal zu berechnenden Zonen pro Anfrage.

In der Diskussion der Simulationsergebnisse werden Begriffe verwendet, die an dieser Stelle eingeführt werden.

**Bearbeitungspeer** ist an einer Abarbeitung einer Anfrage beteiligter Peers,

der in seiner lokalen Datenbanken nach den Daten sucht, die mit der Anfrage spezifiziert wurden.

**Datenpeer** ist Bearbeitungspeers, der tatsächlich die in einer Anfrage spezifizierten Daten in seiner lokalen Datenbank findet. Damit gilt die Beziehung: Datenpeers $\subseteq$ Bearbeitungspeers.

**Weiterleitungspeer** leitet die Suche nach dem für einen Chord-Bezeichner zuständigen Peer an einen anderen Peer weiter. Dazu bedient er sich der Finger-Tabelle.

**Berechnungspeer** ist entweder der Urheber einer Anfrage oder an der verteilten Berechnung der Bezeichner einer Anfrage beteiligt.

**Hop** ist ein logischer Sprung zwischen zwei Knoten, der unabhängig von der darunterliegenden Netzwerkinfrastruktur ist. In der Simulation werden die Hops im Chord-Ring ermittelt, was einer Verbindung zwischen zwei Weiterleitungspeers entspricht.

**Latenz** wird in Hops angegeben und bezeichnet die maximale Länge einer Weiterleitungskette beziehungsweise den längsten Pfad im Weiterleitungsbaum vom Urheber einer Anfrage bis zu einem Bearbeitungspeer.

**Nachrichtengröße** wird in Clustern angegeben, wobei eine Zone als ein Cluster mit demselben Anfangs- und Endbezeichner angesehen wird. Während andere Nutzdaten einer Nachrichten, wie Anfrage und momentane Annäherungsstufe, eine konstante Größe bei der Abarbeitung einer Anfrage aufweisen, beeinflussen die Cluster die Größe einer Nachricht maßgeblich.

**Rechenlast** wird in Zonen angegeben und beschreibt, mit welchem Rechenaufwand ein Berechnungspeer belastet wird, wenn er die Bezeichner für eine Anfrage berechnet.

Sofern nicht anders angegeben, wurden für die Simulation folgende Parameterwerte verwendet. Die Netzwerkgröße $N$ geht von ein Hundert bis Zehntausend. Die Schrittweite wurde für den Exponenten definiert, damit nicht zu viele Durchläufe notwendig sind, um Ergebnisse für eine kleine und sehr große Anzahl von Peers zu erhalten. Die Anzahl an Datenobjekten $O$ im Peer-to-Peer-Netz beträgt das Zehnfache der Anzahl der Peers. Dies soll die Situation widerspiegeln, in der manche Peers viele und andere keine Daten zur Verfügung stellen. Dies darf nicht mit der Suche und der dafür vorgesehenen Speicherung von Datenobjekten verwechselt werden, die mit einer

Anfrage gefunden werden können. Hieran sind alle Peers gleichermaßen beteiligt. Die Datenwerte der Datenobjekte werden nach einem Potenzgesetz nämlich dem Zipfian-Gesetz verteilt, wobei es wenige sehr populäre Daten und sehr viele kaum populäre Daten gibt. Eine solche Verteilung wurde in Peer-to-Peer-Netzen nachgewiesen (vergleiche Gish u. a. (2007); Klemm u. a. (2004); Shu u. a. (2005)) und soll hier ebenfalls Verwendung finden. Die Parameter für den Ausgangsgrad $f$ sowie die Baumtiefe $l$ wurden derart definiert, dass nicht mehr als zehn Prozent der Anzahl der Peers Nachrichten pro Anfrage im Netz unterwegs sind. Das heißt, es werden pro Anfragen im Mittel maximal zehn Prozent der Peers beansprucht. In Tabelle 5.2 auf Seite 60 werden alle Parameterwerte übersichtlich aufgelistet.

Da die sich die Netzgröße linear im Exponenten ändert, wurde für die Darstellung der Graphen eine logarithmische Skala auf der horizontalen Achse gewählt. Dies ist beim Betrachten der Graphen zu beachten. Zudem kommt es vor, dass für eine Kurve sehr kleine und für eine andere Kurve sehr große Werte entstehen. Diese lassen sich ebenfalls nur mit einer logarithmischen Skala auf der vertikalen Achse darstellen. In die Graphen wurde deshalb zusätzlich in grauer Farbe die erste Winkelhalbierende gezeichnet, um eine Interpretation bezüglich des Wachstums zu erleichtern.

| **Parameter** | **Wert(e)** | **Beschreibung** |
|---|---|---|
| $N$ | $10^{2,0}, 10^{2,5}, 10^{3,0}, \ldots, 10^{5,0}$ | Netzwerkgröße |
| $O$ | $10 \cdot N$ | Datenobjekte |
| $f$ | 10 | Ausgangsgrad |
| $l$ | $\lfloor \log_{10} N \rfloor - 1$ | maximale Baumtiefe |

Tabelle 5.2: Alle Parameter der Simulation auf einen Blick.

Die nachfolgenden Experimente zeigen die Wirksamkeit der Optimierungskonzepte. Zuerst wird eine Anfrage mit mehreren raumfüllenden Kurven durchgeführt, um zu zeigen, wie gut das Zonenverhältnis arbeitet. Anschließend werden die grundlegende Weiterleitung und die Nachrichtenbeschränkung miteinander verglichen, wobei das Verhalten der Nachrichtenweiterleitung begutachtet wird. Als Abschluss wird der Einfluss der verteilten Berechnung auf die Rechenlast der Peers aufgezeigt.

## 5.1 Experiment 1 — raumfüllende Kurven

Im ersten Experiment werden mehrere raumfüllende Kurven in Bezug auf die Abarbeitung verschiedener Anfragen untersucht. Es werden Anfragen gestellt, die in ihren Attributen genau mit den Attributen einer der Kurven

übereinstimmen. Dies soll zeigen, inwiefern das heuristische Auswahlverfahren dazu geeignet ist, mithilfe des Zonenverhältnisses die optimale Indexstruktur auszuwählen. Die Definition der Anfragen orientiert sich an der Definition der Kurven. Die erste hat die kleinste und die letzte hat die größte Anzahl an Attributen.

| $\mathbf{q_1}$ | $\mathbf{A_1}$ | | $\mathbf{A_3}$ | | $\surd$ | $\star$ | **Z** |
|---|---|---|---|---|---|---|---|
| $\mathbf{K_1}$ | $\surd$ | | $\surd$ | | 2 | 0 | 1,52 |
| $\mathbf{K_2}$ | $\surd$ | $\star$ | | $\star$ | 1 | 2 | 10,55 |
| $\mathbf{K_3}$ | $\surd$ | $\star$ | $\surd$ | $\star$ | 2 | 2 | 1,52 |

Tabelle 5.3: Es sind sowohl die Attribute der ersten Anfrage ($q_1$) aufgelistet als auch wie die raumfüllenden Kurven ($K$) zur dieser passen. $\surd$ markiert eine Übereinstimmung in den Attributen und $\star$ kennzeichnet ein notwendiges Stellvertreterzeichen. Zudem ist das Zonenverhältnis ($Z$) angegeben.

In Tabelle 5.3 auf Seite 61 ist die erste Anfrage definiert und deren Attributskombination wird mit denen der Kurven anschaulich verglichen. Dem Zonenverhältnis nach kommen die erste und dritte Kurve für eine optimale Abarbeitung in Frage. Jedoch hat die erste Kurve die wenigsten Stellvertreterzeichen, welche andeuten, dass für diese Attribute der gesamte Wertebereich angenommen werden muss, weil das Attribut in der Kurve nicht definiert ist. Hieraus folgt, die erste Kurve sollte zusammen mit der dritten Kurven die wenigsten Bearbeitungspeers haben. Zudem hat die erste Kurve die beste Leistungsfähigkeit in Bezug auf die Berechnung der Indexwerte, da sie die kleinste Anzahl an Stellvertreterzeichen hat. Mit anderen Worten ist die erste Kurve die optimale Indexstruktur für diese Anfrage.

Die Simulationsergebnisse werden in vier Graphen präsentiert, welche die Abarbeitung der Anfrage am deutlichsten abbilden, ohne alle Parameter des Systems aufzeigen zu müssen. Zuallererst wird die Anzahl der Bearbeitungspeers sowie der Datenpeers dargestellt. Diese sollen zeigen, inwiefern das heuristische Zonenverhältnis in einem simulierten Peer-to-Peer-Netz zutrifft. Als nächstes wird die Gesamtzahl aller Hops gezeigt und damit die notwendigen Kommunikationsverbindungen. Je weniger Kommunikation stattfinden muss, um so besser passt die Indexstruktur in Bezug auf diese. Die Latenz ist ein Maß für die maximale Dauer, bis eine Anfrage abgearbeitet wurde, und wird ebenfalls gezeigt. Allerdings nur der Durchschnitt der Latenzen aller Bearbeitungspeers, die keine Nachrichten mehr weiterleiten. Dies soll einerseits verhindern, den Wert zu verfälschen, wenn es aufgrund der Bezeichnerverteilung im Netz zu sehr großen Ausreißern kommt und andererseits kann davon ausgegangen werden, dass nicht alle Suchergebnisse von Bedeutung sind. Das

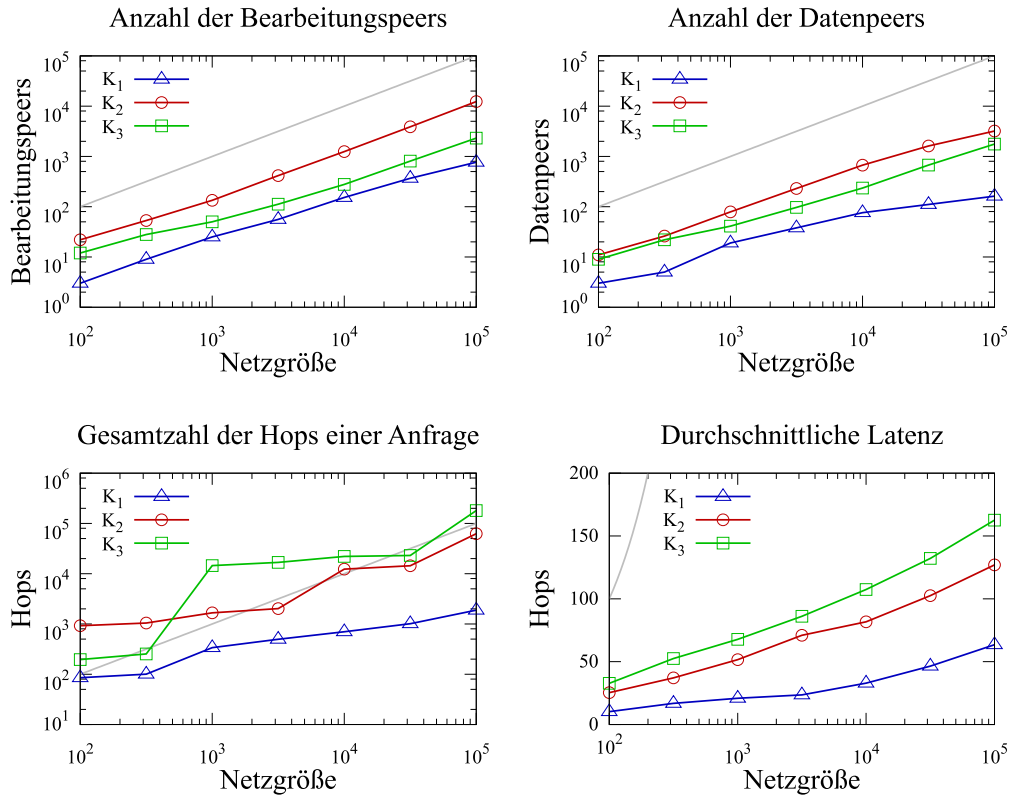



Abbildung 5.1: Es werden Graphen der Simulationsergebnisse für die erste Anfrage $q_1$ gezeigt.

heißt, im Mittel werden bereits fast alle Ergebnisse geliefert sein, bevor die maximale Latenz erreicht wird. Dies sollte wird hier als der Normalfall angenommen. Es sei noch erwähnt, die Latenz steigt mit der Gesamtanzahl der Hops, weil die Latenz die Hop-Zahl der längsten Pfade beinhaltet.

In Abbildung 5.1 auf Seite 62 sind die vier Graphen für die erste Anfrage zu sehen. Es sei nochmals auf die logarithmischen Skalen verwiesen, die hier teilweise zum Einsatz kommen. Es ist klar ersichtlich, der Auswahlalgorithmus wählt die optimale Kurve für diese Anfrage aus. Sowohl die Anzahl der Bearbeitungs- als auch der Datenpeers sind für die erste Kurve am geringsten. Die Zahl dieser Peers steigt mit der Netzgröße linear an, was an der grauen Winkelhalbierenden erkannt werden kann und zu erwarten war. Je mehr Peers in einem Netz vorhanden sind, desto besser sind die Datenobjekte aufgrund der Skalierung der Hilbert-Bezeichner auf die Chord-Bezeichner verteilt. In Bezug auf die Gesamtzahl der Hops und die durchschnittliche Latenz ist die erste Kurve ebenfalls besser. Diese Ergebnisse bestätigen den

Auswahlalgorithmus. Das Zonenverhältnis zeigt sich als gute Metrik für die Abschätzung der Bearbeitungs- und Datenpeers, weil die dritte Kurve im Vergleich zur ersten Kurve zusammen mit dem kleineren Zonenverhältnis auch die geringere Anzahl dieser Peers aufweist. Jedoch scheinen die zwei zusätzlichen Dimensionen der dritten Kurve in Bezug auf die Anfrage eine andere Verteilung der Indexwerte auf dem Chord-Ring zu ergeben, weshalb deutlich mehr Bearbeitungspeers betroffen sind als im Falle der ersten Kurve.

Im Graphen der Gesamtzahl der Hops ist ein merkwürdiges Verhalten der dritten Kurve zu sehen. Dort gibt es ein großen Sprung beim Übergang der Netzgröße von $10^{2,5}$ auf $10^{3,0}$. Dieser lässt sich wie folgt erklären. Da die maximale Baumtiefe, ab der die grundlegende Weiterleitung verwendet wird, mit jedem Hop um Eins dekrementiert wird, kann diese nur ganze Zahlen annehmen. Der Exponente ist jedoch 2,5, weshalb auf 2 abgerundet wird. Daher gibt es erst beim Exponenten 3,0 eine weitere Stufe, in der die Cluster mit $f$ Nachrichten weitergeleitet werden. Nun hat die dritte Kurve zwei Stellvertreterzeichen und damit eine sehr große Anzahl berechneter Zonen. Diese werden einerseits durch die zusätzliche Baumtiefe weiter aufgeteilt und andererseits scheinen die Zonen der Anfrage derart ungeschickt im Chord-Ring verteilt zu sein, dass es zwar zu einer höheren Parallelität aber gleichzeitig auch zu einer höheren Latenz kommt. Bei genauerer Betrachtung des Graphen ist zu sehen, die anderen beiden Kurven haben ebenfalls eine solche Stufe, wenn auch weniger ausgeprägt. Dies scheint mit der Anzahl der Zonen zu tun haben und bedarf einer näheren Untersuchung in zukünftigen Arbeiten.

| $\mathbf{q_2}$ | $\mathbf{A_1}$ | $\mathbf{A_2}$ | | $\mathbf{A_4}$ | $\surd$ | $\star$ | $\mathbf{Z}$ |
|---|---|---|---|---|---|---|---|
| $\mathbf{K_1}$ | $\surd$ | | $\star$ | | 1 | 1 | 10,55 |
| $\mathbf{K_2}$ | $\surd$ | $\surd$ | | $\surd$ | 3 | 0 | 1,23 |
| $\mathbf{K_3}$ | $\surd$ | $\surd$ | $\star$ | $\surd$ | 3 | 1 | 1,23 |

Tabelle 5.4: Es sind sowohl die Attribute der zweiten Anfrage ($q_2$) aufgelistet als auch wie die raumfüllenden Kurven ($K$) zur dieser passen. $\surd$ markiert eine Übereinstimmung in den Attributen und $\star$ kennzeichnet ein notwendiges Stellvertreterzeichen. Zudem ist das Zonenverhältnis ($Z$) angegeben.

Die Übereinstimmung der Attribute zwischen den Kurven und der zweiten Anfrage ist in Tabelle 5.4 auf Seite 63 aufgelistet. Bei der zweiten Anfrage haben erneut zwei Kurven das gleiche Zonenverhältnis. Allerdings hat die zweite Kurve weniger Dimensionen und wird deshalb vom Auswahlalgorithmus als Indexstruktur herangenommen. Die Ergebnisse der Simulation sind in Abbildung 5.2 auf Seite 64 dargestellt.

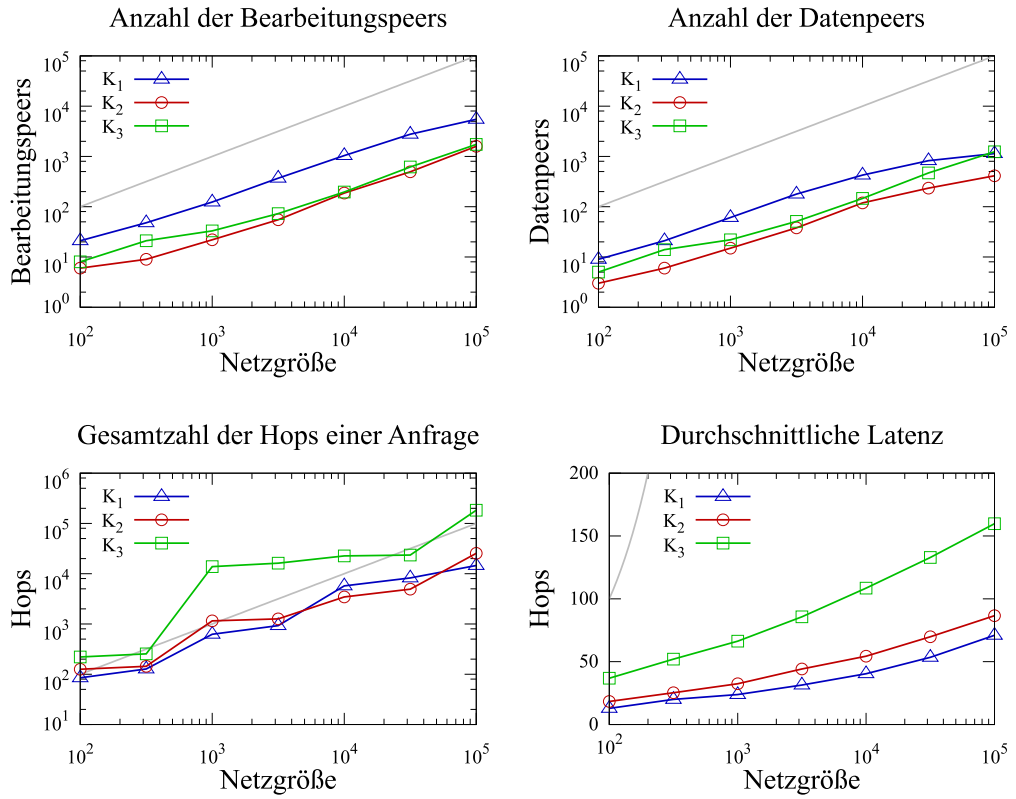



Abbildung 5.2: Es werden Graphen der Simulationsergebnisse für die zweite Anfrage $q_2$ gezeigt.

Auf den ersten Blick ist ersichtlich, die erste Kurve hat eindeutig zu viele Bearbeitungs- und Datenpeers, worauf bereits das Zonenverhältnis hinweist. Mit dieser Kurve müsste eine deutlich größere Anzahl an Peers kontaktiert werden als mit den anderen beiden Kurven. Erstaunlich ist, letztere liegen nahe beieinander, obwohl die dritte Kurve eine Dimension und damit ein Stellvertreterzeichen mehr hat. Hier scheint sich die Vermutung zu zeigen, welche in Kapitel 4.3.1 „Bildung von Attribut-Untermengen“ ab Seite 36 aufgestellt wurde, dass ein Stellvertreterzeichen nur einen linearen Mehraufwand bedeutet. Jedoch zeigt sich sowohl an der Gesamtzahl der Hops als auch an der Latenz die Überlegenheit der zweiten gegenüber der dritten Kurve. Die Anzahl der zu berechnenden Zonen scheint immer noch deutlich zu hoch zu sein, weshalb hier das weiter oben bereits geschilderte Verhalten auftritt.

Als letzter Testfall wird eine Anfrage über alle Attribute der dritten Kurve gestellt, die in Tabelle 5.5 auf Seite 65 zusammen mit den Kurven angegeben ist. Alle Attribute der Kurven stimmen mit der Anfrage überein und

| $\mathbf{q_3}$ | $\mathbf{A_1}$ | $\mathbf{A_2}$ | $\mathbf{A_3}$ | $\mathbf{A_4}$ | $\surd$ | $\star$ | $\mathbf{Z}$ |
|---|---|---|---|---|---|---|---|
| $\mathbf{K_1}$ | $\surd$ | | $\surd$ | | 2 | 0 | 2,84 |
| $\mathbf{K_2}$ | $\surd$ | $\surd$ | | $\surd$ | 3 | 0 | 1,23 |
| $\mathbf{K_3}$ | $\surd$ | $\surd$ | $\surd$ | $\surd$ | 4 | 0 | 0,33 |

Tabelle 5.5: Es sind sowohl die Attribute der dritte Anfrage ($q_3$) aufgelistet als auch wie die raumfüllenden Kurven ($K$) zur dieser passen. $\surd$ markiert eine Übereinstimmung in den Attributen und $\star$ kennzeichnet ein notwendiges Stellvertreterzeichen. Zudem ist das Zonenverhältnis ($Z$) angegeben.

damit gibt es keine unnötigen Stellvertreterzeichen. Wie bereits bei der Herleitung des Zonenverhältnisses gezeigt wurde, bedeutet jedes zusätzliches und übereinstimmendes Attribut ein kleineres Zonenverhältnis. Deshalb wird die dritte Kurve mit den meisten übereinstimmenden Attributen vom Auswahlalgorithmus favorisiert.

Die Ergebnisse können in der Abbildung 5.3 auf Seite 66 betrachtet werden und zeigen erneut, die Kurve mit dem kleinsten Zonenverhältnis hat die geringste Anzahl an Bearbeitungs- und Datenpeers. In diesem Fall ist es die dritte Kurve, wobei die Differenz zu den anderen beiden Kurven nicht groß ist. Dies liegt zum einen an dem kleinen Unterschied im Zonenverhältnis und zum anderen im Mangel an Stellvertreterzeichen. Jedoch zeigt sich im Graphen für die Gesamtzahl der Hops und die Latenz erneut, wie schlecht die dritte Kurve im Vergleich zu den anderen beiden abschneidet. Spätestens bei dieser Anfrage, die alle vier Attribute der dritten Kurve spezifiziert, ist zu sehen, die Anzahl der zu berechnenden Zonen fällt stark ins Gewicht.

Aus Platzgründen und wegen des Auswahlalgorithmus', welcher die verteilte Berechnung nicht berücksichtigt, wurde auf eine Darstellung der Berechnungspeers verzichtet. Allerdings spiegelt sich ihre Anwesenheit in der Gesamtzahl der Hops und der durchschnittlichen Latenz wider. Dies ist besonders bei der dritten Kurve zu sehen, da sich die Werte dieser Graphen über alle drei Anfragen hinweg nicht zu verändern scheinen. In diesem Fall macht die verteilte Berechnung einen zu großen Teil der Kommunikation aus, weshalb der hier vorgestellte Auswahlalgorithmus in Bezug darauf erweitert werden sollte. Da auf die verteilte Berechnung insbesondere bei Kurven mit einer großen Anzahl an Dimensionen nicht verzichtet werden kann, könnte der Auswahlalgorithmus neben dem Zonenverhältnis eine weitere Metrik aufweisen, welche die an der verteilten Berechnung beteiligten Peers entweder berechnet oder abschätzt. Die Anzahl der Berechnungspeers hängt sowohl von den Weiterleitungsparametern $f$ und $l$ als auch der maximal zu berechnenden Zonen ab. Erstere bestimmen, wie oft die Cluster aufgeteilt werden

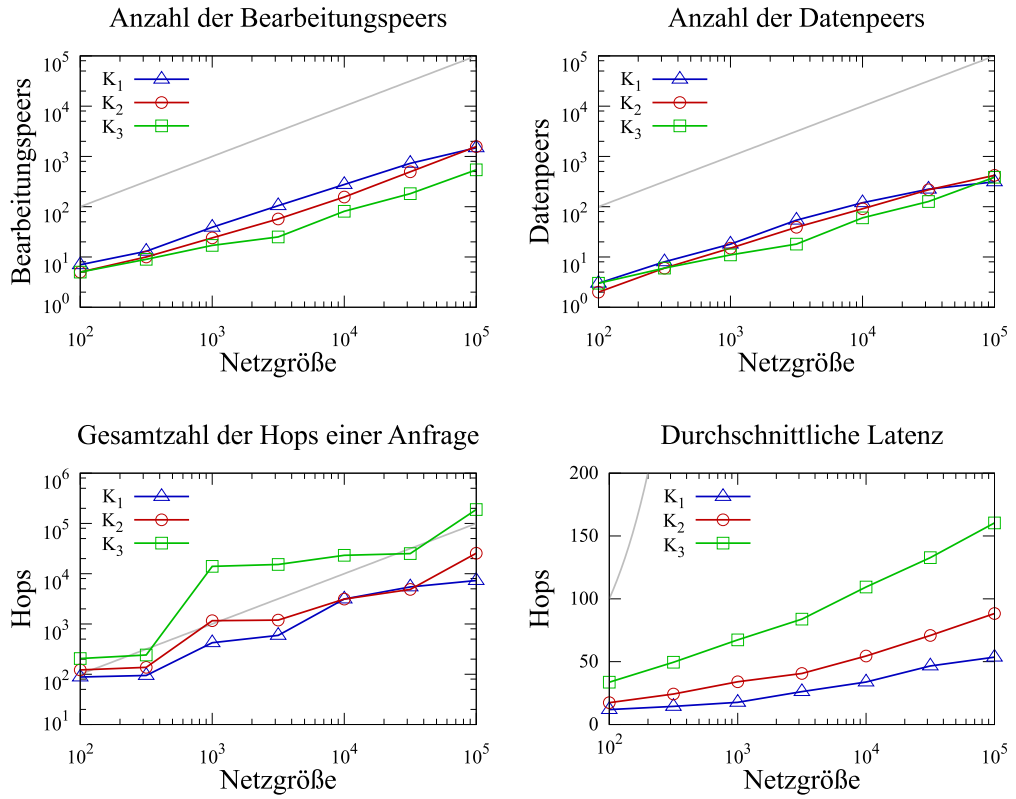



Abbildung 5.3: Es werden Graphen der Simulationsergebnisse für die dritte Anfrage $q_3$ gezeigt.

und damit wie klein sie werden. Letztere geben an, wie stark sich ein Peer an der verteilten Berechnung beteiligt. Dies bedarf weiterer Überlegungen sowie Untersuchungen, die über den Rahmen dieser Diplomarbeit hinausgehen. Daher bleiben sie zukünftigen Arbeiten überlassen.

Allein in Bezug auf die Anzahl der von einer Kurve erzeugten Indexwerte hat sich das Zonenverhältnis als eine gute Metrik erwiesen, obwohl es sich um einen heuristischen Ansatz handelt. Bei allen drei Anfragen konnte von vornherein diejenige Kurve bestimmt werden, welche die geringste Anzahl an Bearbeitungspeers aufwies. Sofern die verteilte Berechnung nicht stärker ins Gewicht fällt als die Kommunikation mit den Bearbeitungspeers, stellt der Auswahlalgorithmus einen optimalen Lösungsansatz dar.

## 5.2 Experiment 2 — Beschränkung der Nachrichten

Nachdem aus dem ersten Experiment geschlussfolgert werden kann, wie gut die Auswahl einer Indexstruktur mit dem heuristischen Auswahlalgorithmus ist und dieser sich als optimal für die gestellten Anforderungen erwies, werden nachfolgend die Eigenschaften der ersten Optimierung, der Nachrichtenbeschränkung bewertet.

Für das zweite Experiment wurde die zweite Hilbert-Kurve (siehe Tabelle 5.1 auf Seite 58) und eine Anfrage mit zwei Attributen der Kurve ausgewählt. Das heißt, es gibt ein Attribut in der Kurve, für dass der gesamte Wertebereich angenommen werden muss. Dies erzeugt eine ausreichend große Anzahl von Zonen, um die grundlegende Weiterleitung und die Nachrichtenbeschränkung zu vergleichen. Bei diesem Experiment wurde die verteilte Berechnung deaktiviert. Die Ergebnisse sind in der Abbildung 5.4 auf Seite 68 zusammengetragen und werden nachfolgend erläutert.

Die Eigenschaften der beiden Weiterleitungen sollten sich in zwei Punkten deutlich widerspiegeln. Zum einen ist die Latenz bei der grundlegenden Weiterleitung sehr klein, da jeder Cluster mit einer eigenen Nachricht an den zuständigen Peer gesendet und von dort aus linear Weitergeleitet wird. Zum anderen ist die Anzahl der Nachrichten pro Weiterleitungsstufe bei der Nachrichtenbeschränkung durch den Ausgangsgrad und die maximale Baumtiefe fest vorgegeben, weshalb bei einer großen Anzahl von Clustern, letztere deutlich weniger Nachrichten pro Stufe haben sollte. Zudem gibt es bei der grundlegenden Weiterleitung nur einige wenige Stufen, in der alle Cluster-Nachrichten versendet werden.

Im Graphen mit dem Nachrichtenaufkommen pro Baumtiefe wird das Nachrichtenaufkommen über alle Weiterleitungsstufen gemittelt dargestellt. Es ist ersichtlich, die grundlegende Weiterleitung zeigt eine sehr große Anzahl von Nachrichten pro Stufe, da die Anzahl der Nachrichten der Anzahl der Cluster entspricht und es nur sehr wenige Stufen gibt. Bei einer sehr großen Anzahl von Peers, werden die Werte etwas kleiner, weil jeder Peer für weniger Bezeichner verantwortlich ist und die lineare Weiterleitung die Baumtiefe erhöht. Die Wirkung der Nachrichtenbeschränkung zeigt sich dagegen in den deutlich niedrigeren Werten. Das heißt, es werden nur eine geringe Anzahl von Nachrichten pro Stufe versendet. Der Anstieg der Werte ergibt sich aus dem Simulationsaufbau, bei dem die maximale Baumtiefe so gewählt ist, dass die maximale Anzahl der parallelen Nachrichten zehn Prozent der Netzgröße nicht übersteigt. Das Nachrichtenaufkommen wächst folglich mit der Größe des Netzes.

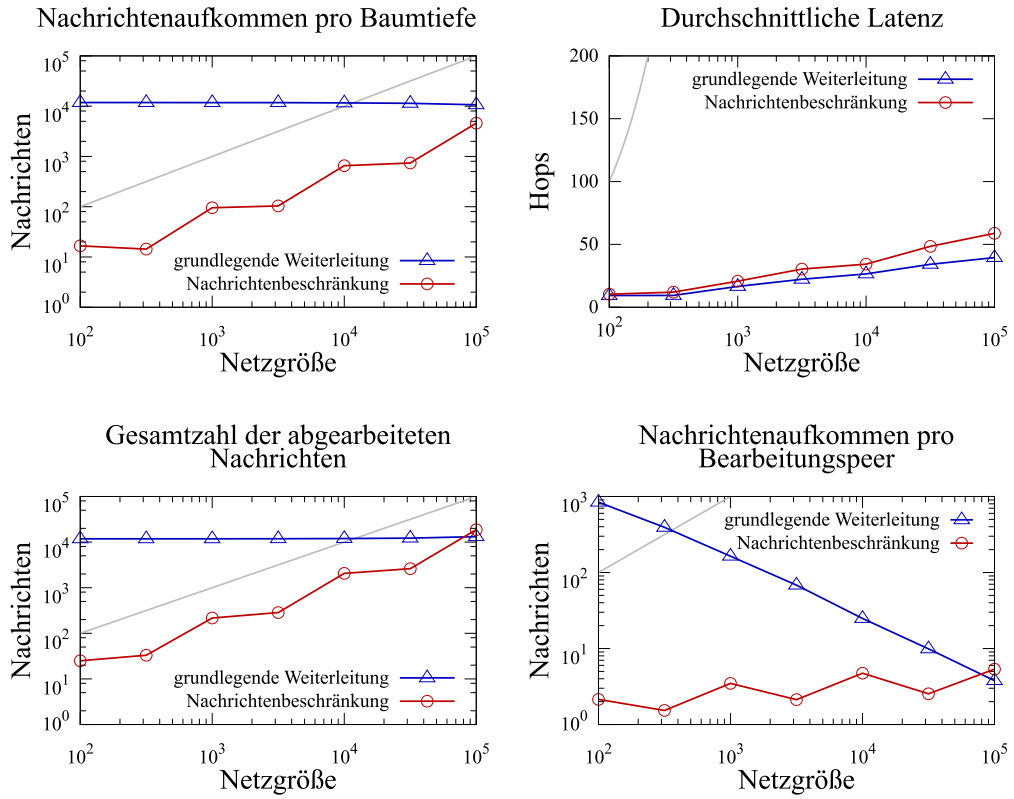



Abbildung 5.4: Es werden Graphen der Simulationsergebnisse für den Vergleich der grundlgenden Weiterleitung mit der Nachrichtenbeschränkung gezeigt.

Die Beschränkung der maximalen Anzahl paralleler Nachrichten sollte zu einer erhöhten Latenz führen. Dies kann im Graphen der durchschnittlichen Latenz abgelesen werden. Da die parallele Nachrichtenanzahl der Beschränkung mit der Netzgröße erhöht wird, sollte sich die Latenz der Beschränkung an die der grundlegenden Weiterleitung annähern. Dies ist jedoch nicht der Fall und hat mehrere Ursachen. Die zu versendenden Cluster werden bei der Beschränkung in $f$ Nachrichten aufgeteilt. Mit einer kleinen Netzgröße, ist ein Peer für einen großen Bezeichnerbereich zuständig, weshalb diese $f$ Nachrichten sehr günstig sind, wie dem Graphen des Nachrichtenaufkommens pro Bearbeitungspeer entnommen werden kann. Mit der wachsenden Anzahl an Peers im Netz sinkt der Bezeichnerbereich eines jeden Peers. Damit erhöht sich die Weiterleitungskette, auch wenn mehr parallele Nachrichten versendet werden, was wiederum die Latenz erhöht. Zudem wird die maximale Parallelität erst mit der maximalen Baumtiefe erreicht. Bei einem entsprechend

kleinen Ausgangsgrad $f$ kann dies eine sehr große Baumtiefe erfordern. Dies erhöht die Latenz zusätzlich. Ferner werden die Cluster bei der Aufteilung in $f$ Nachrichten zusammengefasst, wodurch die Weiterleitung ab der maximalen Baumtiefe unnötig lange Weiterleitungsketten erzeugen kann. Letzteres kann im Graphen des Nachrichtenaufkommens pro Bearbeitungspeer erkannt werden. Die Anzahl der Nachrichten pro Peer steigt im Falle der Nachrichtenbeschränkung nur sehr langsam mit der Netzgröße an, obwohl die Gesamtzahl der abgearbeiteten Nachrichten linear mit der Netzgröße wächst. Dies lässt darauf schließen, eine Zusammenfassung der Cluster wirkt sich günstig auf das Nachrichtenaufkommen eines Peers aus, wobei sich die Latenz aufgrund von längeren Weiterleitungsketten erhöht. Im Falle der grundlegenden Weiterleitung ist das Nachrichtenaufkommen bei kleinen Netzgröße bedenklich hoch, weil jeder Peer einen sehr großen Bezeichnerbereich verwaltet und mehrere Cluster in diese fallen. Mit steigender Netzgröße und damit sinkender Bezeicherbereiche verringert sich allerdings das Nachrichtenaufkommen. Hier sei jedoch angemerkt, es sind nähere Untersuchungen nötig, um festzustellen, ob es keine ungünstigen Bereiche einer Anfrage gibt, die diese Eigenschaft nicht aufweisen.

Die Ergebnisse belegen, eine Nachrichtenbeschränkung ist aus zweierlei Hinsicht sinnvoll. Zum einen aufgrund ihrer Definition, die maximale Anzahl an parallelen Nachrichten zu beschränken, was sich im Nachrichtenaufkommen pro Baumtiefe niederschlägt. Zum anderen werden Cluster zusammengefasst, die alle in den Bereich eines Peers fallen, was das Nachrichtenaufkommen eines jeden Peers sehr gering hält. Zwar steigt die Latenz im Vergleich zur grundlegenden Weiterleitung an, doch ist die Differenz vernachlässigbar im Verhältnis zum Nutzen.

## 5.3 Experiment 3 — verteilte Berechnung

Während das Verhalten der ersten Optimierung im vorherigen Kapitel anhand der Simulationsergebnisse bewertet wurde, findet in diesem Kapitel eine Begutachtung der zweiten Optimierung statt. Hierbei handelt es sich um das Verteilen der Berechnung aller Zonen, die für eine Anfrage berechnet werden müssen. Die Berechnung wird für jeden Peer künstlich auf eine maximale Anzahl von Zonen beschränkt, um eine unnötige Wartezeit beim Urheber der Anfrage zu vermeiden.

Für die Simulation wurde die Netzgröße auf 1.000 Peers festgelegt und die maximale Annäherungsstufe der zweiten Kurve schrittweise von 3 auf 9 erhöht. Der Exponent für die maximale Anzahl zu berechnender Zonen namens $B$ wurde von 16 auf 9 gesenkt, damit die Berechnung früher verteilt

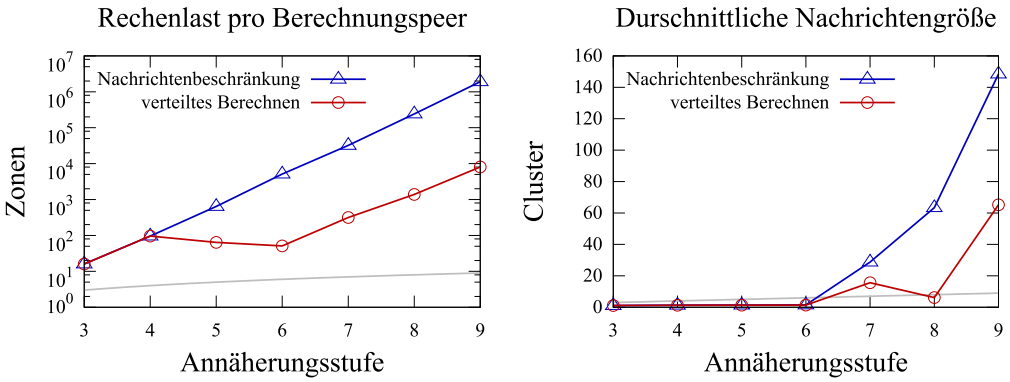



Abbildung 5.5: Es werden Graphen der Simulationsergebnisse für die Bewertung der verteilten Berechnung gezeigt.

wird. Die Anfrage wurde aus Experiment 2 übernommen. Die Anfrage wurde einmal mit deaktivierter und einmal mit aktivierter verteilten Berechnung durchgeführt.

Zuallererst ist die Rechenlast pro Bearbeitungspeer interessant, welche im Graphen Rechenlast pro Bearbeitungspeer gezeigt wird. Diese findet sich zusammen mit dem anderen Graphen zur durchschnittlichen Nachrichtengröße in Abbildung 5.5 auf Seite 70. Aufgrund der Berechnungsformel für die Anzahl der Zonen in einer Hilbert-Kurve $2^{k \cdot d}$ wächst die Anzahl der Zonen der Anfrage in Bezug auf $k$ exponentiell. Das heißt, ohne verteilte Berechnung muss ein einzelner Peer die gesamte Rechenlast tragen, was zu einer sehr langen Berechnungszeit führen kann. Im Graphen ist die Abarbeitung nur mit der Nachrichtenbeschränkung als Optimierung zu sehen, welche stetig exponentiell wächst, was erwartet wurde. Es sei auf die logarithmische Skala der vertikalen Achse hingewiesen, weshalb das Wachstum auf den ersten Blick linear erscheint. Zudem ist an der eingezeichneten Winkelhalbierenden erkennbar, dass die anderen beiden Kurven deutlich stärker wachsen. Das Absinken der Kurve für die verteilte Berechnung bei der Annäherungsstufe 5 zeigt, dass der ursprüngliche Peer seine festgelegte Rechenkapazität überschreiten würde und deshalb die Berechnung verteilt hat. Da mehr Peers an der Berechnung beteiligt sind, fällt die Rechenlast pro Peer. Bei der Annäherungsstufe 6 sinkt der Wert sogar noch weiter. Dies ist ein Hinweis für noch mehr Berechnungspeers, die gemeinsam an der Berechnung der Zonen arbeiten. Anschließend wachsen beide Kurven wieder. Das weitere Wachstum der Kurve ohne verteile Berechnung ist durch die Berechnungsformel begründet und wird stetig weitergehen. Für die optimierte Kurve bleibt zu klären, ob es wieder zu einer Absenkung kommen wird. Hierbei ist die Bedingung zu

beachten, dass jeder Peer zumindest die nächste Annäherungsstufe berechnen muss. Deshalb berechnet jeder Peer ab der 7. Annäherungsstufe mehr Zonen, als mit die Beschränkung der verteilten Berechnung definiert wurde. Dies wird durch den zweiten Graphen mit der durchschnittlichen Nachrichtengröße bestätigt. Die Anzahl der Cluster pro Nachricht steigt beim verteilten Berechnen, wenn auch verzögert, exponentiell (beide Skalen sind normal). Mit anderen Worten die ursprüngliche Anzahl der Zonen am Startknoten ist bereits derart groß, dass die festgelegte Rechenlast deutlich überschritten ist, der Peer jedoch mindestens die Zonen der nächsten Annäherungsstufe berechen muss. Diese Zonen werden in $f$ Nachrichten aufgeteilt und an andere Peers zur weiteren Berechnung weitergeleitet. Jedoch ist die Anzahl der enthaltenen Zonen bereits so groß, dass jeder Peer wie der Urheber der Anfrage verfahren wird, bis entweder die maximale Annäherungsstufe erreicht ist oder die Cluster beim Empfang eine Anzahl von Zonen beinhalten, die nicht die spezifizierte Rechenlast eines Peers überschreitet.

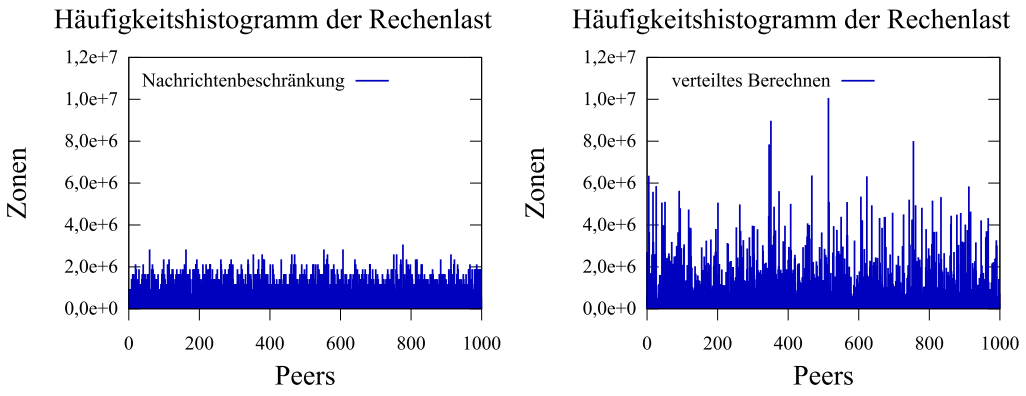



Abbildung 5.6: Die Graphen zeigen Histogramme der Rechenlast von Peers im Netz. Wobei die verteilte Berechnung einmal deaktiviert (links) und einmal aktiviert ist (rechts).

Die Simulationsergebnisse zeigen, für die Verteilung der Berechnung ist nicht nur eine Begrenzung der Rechenlast pro Peer erforderlich. Die Weiterleitungsparameter der Nachrichtenbeschränkung spielen auch eine entscheidene Rolle. Je größer der Ausgangsgrad, desto kleiner wird die Anzahl an Zonen, welche mit den $f$ Nachrichten versendet werden. Die maximale Baumtiefe ist ebenfalls von Bedeutung. Da die Anzahl der Zonen mit jeder weiteren Annäherungsstufe exponentiell wächst, wäre eine Weiterleitung mit exponentiellem Wachstum der Nachrichten notwendig, um die Nachrichtengröße zu beschränken. Dies ist mit einer sehr großen maximalen Baumtiefe gegeben, da die erreichte Anzahl an parallelen Nachrichten durch die Berechnungsfor-

mel $f^l$ begrenzt wird, welche ebenfalls ein exponentielles Wachstum aufweist. Allerdings führt dies zu einer Nachrichtenmenge, die in Bezug zu der Anzahl der Zonen und damit Cluster steht. Das widerspricht der Grundannahme der Nachrichtenbeschränkung, wie sie in dieser Arbeit gegeben ist (vergleiche Kapitel 4.5.3 ab Seite 49).

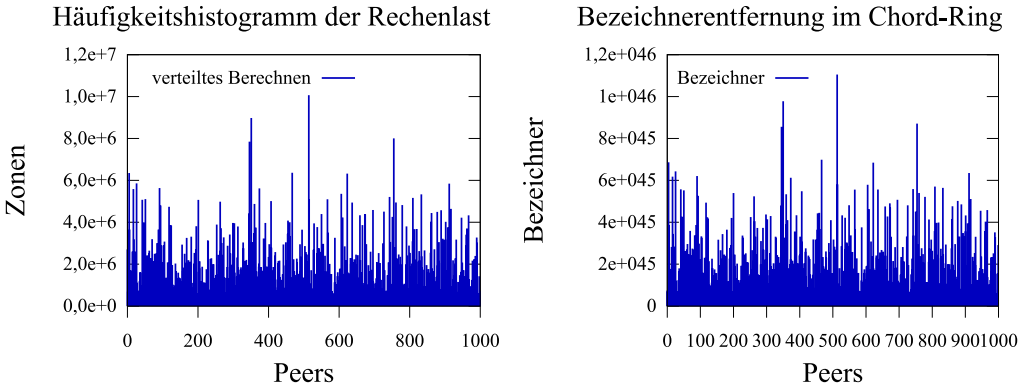



Abbildung 5.7: Hier werden das Häufigkeitshistogramm für die Rechenlast sowie die Bezeichnerbereiche der Peers gegenübergestellt. Die Bereiche werden angegeben als die Entfernung zweier Bezeichner.

Ein wesentlicher Aspekt der verteilten Berechnung ist, welche Peers daran beteiligt werden. Ein Ansatz ist, die Berechnung von Peers durchführen zu lassen, die auch für die enthaltenen Zonen verantwortlich sind. Auf diese Weise wären die Bezeichner schneller bei ihrem Verwalter. Für sehr populäre Daten würde der verantwortliche Peer allerdings nicht nur mit einer höheren Anfragequote, sondern zusätzlich auch mit Berechnungen belastet. Daher wird die Berechnung rein zufällig im Netz verteilt. In Abbildung 5.6 auf Seite 71 sind zwei Histogramme zusehen, wobei die verteilte Berechnung beim rechten zum Einsatz kommt und beim linken nicht. Hierfür wurden die oben beschriebenen Anfragen 5.000 mal von zufälligen Knoten aus durchgeführt. Im Falle ohne verteilte Berechnung ist eine Gleichverteilung der Rechenlast zu sehen. Wird dagegen die verteilte Berechnung verwendet, scheint es keine gute Verteilung zu geben. Hierzu sollte erwähnt werden, eine Anfrage startet immer an einem Peer, weshalb die Lastverteilung ausgegelichen ist. Bei einer verteilten Berechnung wird hingegen ein Bezeichner zufällig gewählt, an den die Anfrage zur Berechnung weitergeleitet wird. Wird das Histogramm für die verteilte Berechnung unter der Annahme gesehen, der verwendete Pseudozahlenzufallsgenerator erzeugt eine ausreichend gute Gleichverteilung, so spiegeln sich im Histogramm die Bezeichnerbereich der einzelnen Peers wider. Diese Annahme trifft tatsächlich zu, wie die Gegenüberstellung des Hi-

stogramms mit den Bezeichnerbereichen in der Abbildung 5.7 auf Seite 72 bestätigt. Damit entsteht zwar eine Gleichverteilung der Berechnung allerdings nach Bezeichner und nicht nach Peers. Besitzt ein Peer einen großen Bezeichnerbereich erhält er ebenfalls eine große Anzahl an Berechnungsanfragen.

Wie sich im ersten Experiment gezeigt hat, führt die Verteilung der Berechnung zu einer größeren Gesamtzahl der Hops und letztendlich auch zu einer höheren Latenz. Hinzu kommt die Beschränkung, mindestens die nächste Annäherungsstufe berechnen zu müssen. Zukünftige Untersuchungen sollten einerseits versuchen, die beiden Optimierungen zu verschmelzen, so dass die Nachrichtenbeschränkung der verteilten Berechnung entgegenkommt, sowie andererseits die Berechnung zu verfeiern, damit ein Peer seine definierte maximale Rechenlast einhalten kann. Die Verteilung der Berechnung orientiert sich an den zugewiesenen Bezeichnerbereichen im Chord-Ring. Sind diese nicht gleichverteilt, werden Peers mit der Verantwortlichkeit für mehr Bezeichner in Bezug auf die verteilte Berechnung benachteiligt. Dies wirkt allerdings einem Lastausgleich entgegen, der populäre Bezeichnerbereiche über mehrere Knoten streut und weniger populäre Bezeichnerbereiche auf Peers konzentriert. Die Rechenlast wird auf den Peers mit den populären Bereichen durch die Anfragen an sich erzeugt und auf den Peers mit weniger populären Anfragen durch verteilte Berechnungen.

# Kapitel 6

# Zusammenfassung

Peer-to-Peer-Netze sind in den letzten Jahren zu flexiblen, skalierbaren und effizienten Systemen herangereift, die in dynamischen Netzen für eine effiziente Suche eingesetzte werden. Es wurden Ansätze entwickelt, die multidimensionale Bereichsanfrage ermöglichen. Die Aufgabe dieser Diplomarbeit ist die Optimierung solcher multidimensionaler Bereichsanfragen, die auf raumfüllenden Kurven und verteilten Hash-Tabellen basieren. Die bisherigen Entwicklungen auf diesem Gebiet bilden Indexstrukturen über alle Attribute im jeweiligen System und bieten Bereichsanfragen mit weniger Attributen daher keinen optimalen Index. Es wurde gezeigt, eine solche Lösung ist mit einem Mehraufwand in Bezug auf die zu kontaktierenden Peers verbunden.

Die Optimierung besteht darin, Anfragen einen optimalen Index zu bieten. Hierfür werden mittels einer Heuristik Attributskombinationen gebildet. Aus diesen muss die für eine Anfrage am besten passende Kombination ausgewählt werden. Dafür wurde sowohl eine Metrik namens Zonenverhältnis als auch ein heuristischer Auswahlalgorithmus vorgestellt und theoretische begründet. Zudem wurden zwei weitere Optimierungen eingeführt, welche die maximale Anzahl von Nachrichten im Peer-to-Peer-Netz begrenzen sowie aufwändige Berechnungen verteilen. Eine Evaluation wurde mithilfe einer Simulation durchgeführt.

Die Ergebnisse der Simulation bestätigen die Wirkungsweise der Attributskombinationen in Bezug auf die Anzahl der an der Abarbeitung einer Anfrage beteiligten Peers. Damit stellt die hier vorgestellte Optimierung eine optimale Lösung dar. Jedoch hat sich gezeigt, kann dies durch eine große Anzahl an zu berechnenden Zonen getrübt werden, da die Berechnungspeers einen größeren Teil an der Bearbeitung ausmachen können als die Bearbeitungspeers. Daher ist ist die hier vorgestellte Optimierung allein nicht ausreichend.

Die Nachrichtenbeschränkung, mit der die maximale Anzahl an parallelen

Nachrichten im Netz garantiert wird, hat sich als eine effektive Optimierung herausgestellt, auch wenn die Latenz etwas ansteigt. Denn letzterer Umstand ist im Vergleich zum Nutzen akzeptierbar.

Die verteilte Berechnung dagegen hat einen großen Einfluss auf die Gesamtzahl der Hops und damit auch auf die Latenz. Jedoch sind diese Effekte tolerierbar, wenn bedacht wird, dass ein Peer sehr lange wenn nicht sogar deutlich länger für die alleinige Berechnung aller Zonen braucht. Die damit eingeführte Beschränkung der maximal zu berechnenden Zonen, wird durch die künstlich eingeführte Notwendigkeit, mindestens die nächste Annäherungsstufe zu berechnen für eine große Anzahl von Clustern unwirksam, wie sich im dritten Experiment herausgestellt hat.

Alles in allem wurde deutlich gezeigt, eine Menge von kleinen raumfüllenden Kurven ist effizienter als eine sehr große Kurve. Allerdings bedeutet dies einen größeren Speicherbedarf im Netz, welcher jedoch tolerierbar ist. Für die Bildung der Attributskombinationen wurde eine Heuristik vorgestellt und der präsentierte Auswahlalgorithmus arbeitet optimal.

# Kapitel 7

# Ausblick

In zukünftigen Arbeiten sollte zuallererst die verteilte Berechnung zusammen mit der Nachrichtenbeschränkung überdacht werden. Insbesondere raumfüllende Kurven mit einer großen Zahl an Dimensionen machen eine Berchnung einer sehr großen Anzahl von Clustern notwendig. Hier ist definitiv eine weitere Optimierung notwendig. Hierzu gehört auch eine Berechnung ohne den Zwang, die zumindest die nächste Annäherungsstufe berechnen zu müssen, wobei darauf geachtet werden muss, Nachrichten nicht ewig im Chord-Ring weiterleiten zu lassen.

Der größere Speicherverbrauch des Systems kann für eine höhere Ausfallsicherheit genutzt werden. Während andere Systeme explizit Redundanz einfügen müssen, ist diese im Systemmodell dieser Arbeit implizit vorhanden. Es bedarf allerdings einer Strategie, wie dies Redundanz genutzt wird.

Die vorgestellte Heuristik ist im Vergleich zu den Einflussfaktoren simpel gestatltet. Hier sind weitere Untersuchungen notwendig, wie sich verschiedene Attributskombinationen gegenseitig beeinflussen, um eine Menge an hochwertigeren Attributskombinationen zu erhalten, die für möglichst viele Anfragen eine optimale Kombination bieten.

Die Zonen der raumfüllenden Kurven werden zwar aufgrund der Skalierung der Kurve auf den Chord-Ring gleichmäßig verteilt, allerdings wird dabei nicht die Verteilung der Datenwerte berücksichtigt. Hierfür ist ein Lastausgleich erforderlich, welcher ebenfalls andere Bereiche des Systemmodells betrifft. Dies wurde an den entsprechenden Stellen in der Ausarbeitung vermerkt.

# Anhang A

# Beweisführungen

## A.1 Anzahl der Attributskombinationen

Die Formel aus Kapitel 4.3.1 „Bildung von Attribut-Untermengen“ lässt sich wie folgt herleiten:

$$\forall n, k \in \mathbf{N} : S = \sum_{k=2}^{n} \binom{n}{k} = 2^n - n - 1.$$

Es wird die Summe aller Kombinationen für $k$ bis $n$ Attribute mithilfe des Binomialkoeffizienten berechnet. Die Formel ganz rechts lässt sich entweder durch Berechnung mehrerer Werte entdecken oder berechnen, wenn an eine Potenzmenge gedacht wird. Ein Attribut ist entweder vorhanden oder nicht. Das sind zwei Möglichkeiten. Bei n Attributen ergeben sich insgesamt $2 \cdot 2 \cdot \ldots \cdot 2 = 2^n$ Möglichkeiten. Wird die Anzahl der Möglichkeiten für die leere Menge sowie eine einelementige Menge abgezogen ergibt sich die Berechnungsformel $2^n - n - 1$. □

## A.2 Mehrfach belegte Bezeichner

Die Anzahl der überlappenden Indices bei der direkten Abbildung ($2^{k \cdot d} \bmod 2^m$) ist $2^{k \cdot d} - 2^m$. Die Anzahl der überlappenden Indices bei der Skalierung ($s \cdot 2^{k \cdot d}$) ist ebenfalls $2^{k \cdot d} - 2^m$, da die Anzahl der resultierenden Indices von der Gesamtzahl abgezogen werden muss. Dies zeigt, bei beiden Abbildungen handelt es sich um die gleiche Anzahl an überlappenden Indices. □

## A.3 Maximale Anzahl der parallelen Nachrichten

Die maximale Anzahl der parallelen Nachrichten für einen Ausgangsgrad $f$ und eine maximale Baumtiefe $l$ lässt sich wie folgt berechnen. Der erste Peer sendet $f$ Nachrichten. Es wird angenommen, ein Peer erhält höchsten eine Nachricht. Damit ergeben sich $f$ Peers, die wiederum $f$ Nachrichten senden, womit es bereits $f \cdot f = f^2$ Nachrichten sind. In der nächsten Stufe versenden $f^2$ Peers erneut $f$ Nachrichten. Dies ergibt $f^2 \cdot f = f^3$ als maximale Anzahl an Nachrichten. Dies wird soweit fortgeführt, bis eine Baumtiefe von $l$ und damit eine maximale Anzahl an parallelen Nachrichten von $f^l$ erreicht ist. □

# Abbildungsverzeichnis

# Literaturverzeichnis

[Andrzejak und Xu 2002] Andrzejak, Artur ; Xu, Zhichen: Scalable, Efficient Range Queries for Grid Information Services. In: *P2P '02: Proceedings of the Second International Conference on Peer-to-Peer Computing*. Washington, DC, USA : IEEE Computer Society, 2002, S. 33. – ISBN 0-7695-1810-9

[Bharambe u. a. 2004] Bharambe, A. ; Agrawal, M. ; Seshan, S.: Mercury: Supporting Scalable Multi-Attribute Range Queries. In: *Proceedings of the SIGCOMM Symposium on Communications Architectures and Protocols*. Portland, OR, Aug 2004

[Butz 1971] Butz, A. R.: Alternative Algorithm for Hilbert's Space-Filling Curve. In: *IEEE Trans. Comput.* 20 (1971), Nr. 4, S. 424–426. – ISSN 0018-9340

[Cai u. a. 2003] Cai, Min ; Frank, Martin ; Chen, Jinbo ; Szekely, Pedro: MAAN: A Multi-Attribute Addressable Network for Grid Information Services. In: *GRID '03: Proceedings of the Fourth International Workshop on Grid Computing*. Washington, DC, USA : IEEE Computer Society, 2003, S. 184. – ISBN 0-7695-2026-X

[Datta u. a. 2005] Datta, Anwitaman ; Hauswirth, Manfred ; John, Renault ; Schmidt, Roman ; Aberer, Karl: Range Queries in Trie-Structured Overlays. In: *P2P '05: Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing*. Washington, DC, USA : IEEE Computer Society, 2005, S. 57–66. – ISBN 0-7695-2376-5

[Dürr 2007] Dürr, Frank: *Konzepte der Peer-to-Peer-Systeme*. Vorlesung Sommersemester 2007, Institut für Parallele und Verteilte Systeme, Universität Stuttgart. 2007

[Ganesan u. a. 2004] Ganesan, Prasanna ; Yang, Beverly ; Garcia-Molina, Hector: One torus to rule them all: multi-dimensional queries in

P2P systems. In: *WebDB '04: Proceedings of the 7th International Workshop on the Web and Databases*. New York, NY, USA : ACM, 2004, S. 19–24

[Gish u. a. 2007] Gish, Adam S. ; Shavitt, Yuval ; Tankel, Tomer: Geographical Statistics and Characteristics of P2P query strings. In: *IPTPS '07: Online Proceedings of 6rd International Workshop on Peer-to-Peer Systems*, 2007

[Gotsman und Lindenbaum 1996] Gotsman, C. ; Lindenbaum, M.: *the metric properties of discrete space-filling curves*. 1996. – URL `citeseer.ist.psu.edu/gotsman96metric.html`

[Hilbert 1891] Hilbert, David: Über die stetige Abbildung einer Linie auf ein Flächenstück. In: *Mathematische Annalen*, 1891, S. 459–460

[Karger u. a. 1997] Karger, David ; Lehman, Eric ; Leighton, Tom ; Panigrahy, Rina ; Levine, Matthew ; Lewin, Daniel: Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web. In: *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. New York, NY, USA : ACM, 1997, S. 654–663. – ISBN 0-89791-888-6

[Kemper und Eikler 2001] Kemper, Alfons ; Eikler, André: *Datenbanksysteme: eine Einführung*. 4. überarb. und erw. München : Oldenburg, 2001

[Kesselman und Foster 1998] Kesselman, Carl ; Foster, Ian: *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, November 1998. – ISBN 1558604758

[Klemm u. a. 2004] Klemm, Alexander ; Lindemann, Christoph ; Vernon, Mary K. ; Waldhorst, Oliver P.: Characterizing the query behavior in peer-to-peer file sharing systems. In: *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA : ACM, 2004, S. 55–67. – ISBN 1-58113-821-0

[Lawder 2000] Lawder, J.: *Calculation of Mappings between One and n-dimensional Values Using the Hilbert Space-Filling Curve*. 2000. – URL `citeseer.ist.psu.edu/lawder00calculation.html`

[Lv u. a. 2002] Lv, Qin ; Cao, Pei ; Cohen, Edith ; Li, Kai ; Shenker, Scott: Search and replication in unstructured peer-to-peer networks. In:

*ICS '02: Proceedings of the 16th international conference on Supercomputing*. New York, NY, USA : ACM, 2002, S. 84–95. – ISBN 1-58113-483-5

[Mahlmann und Schindelhauer 2007] Mahlmann, Peter ; Schindelhauer, Christian: *Peer-to-Peer-Netzwerke: Algorithmen und Methoden*. Berlin, Heidelberg : Springer-Verlag, 2007

[Mokbel u. a. 2003] Mokbel, Mohamed F. ; Aref, Walid G. ; Kamel, Ibrahim: Analysis of Multi-Dimensional Space-Filling Curves. In: *Geoinformatica* 7 (2003), Nr. 3, S. 179–209. – ISSN 1384-6175

[Moon u. a. 2001] Moon, Bongki ; Jagadish, H. v. ; Faloutsos, Christos ; Saltz, Joel H.: Analysis of the Clustering Properties of the Hilbert Space-Filling Curve. In: *IEEE Trans. on Knowl. and Data Eng.* 13 (2001), Nr. 1, S. 124–141. – ISSN 1041-4347

[PeerSim 2006] PeerSim: *PeerSim: A Peer-to-Peer Simulator*. 2006. – URL `http://peersim.sourceforge.net`

[Ramabhadran u. a. 2004] Ramabhadran, Sriram ; Hellerstein, Joseph ; Ratnasamy, Sylvia ; Shenker, Scott: Prefix Hash Tree - An Indexing Data Structure over Distributed Hash Tables. In: *PODC 2004: Twenty-Third Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, 2004

[Ripeanu 2001] Ripeanu, Matei: *Peer-to-Peer Architecture Case Study: Gnutella Network*. 2001. – URL `citeseer.ist.psu.edu/ripeanu01peertopeer.html`

[Sagan 1994] Sagan, Hans: *Space-Filling Curves*. New York, Berlin, Heidelberg, London : Springer-Verlag, 1994

[Saroiu u. a. 2003] Saroiu, Stefan ; Gummadi, Krishna P. ; Gribble, Steven D.: Measuring and analyzing the characteristics of Napster and Gnutella hosts. In: *Multimedia Syst.* 9 (2003), Nr. 2, S. 170–184. – ISSN 0942-4962

[Schmidt und Parashar 2003] Schmidt, Cristina ; Parashar, Manish: Flexible Information Discovery in Decentralized Distributed Systems. In: *HPDC '03: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*. Washington, DC, USA : IEEE Computer Society, 2003, S. 226. – ISBN 0-7695-1965-2

[Shu u. a. 2005] Shu, Yanfeng ; Ooi, Beng C. ; Tan, Kian-Lee ; Zhou, Aoying: Supporting Multi-Dimensional Range Queries in Peer-to-Peer Systems. In: *P2P '05: Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing*. Washington, DC, USA : IEEE Computer Society, 2005, S. 173–180. – ISBN 0-7695-2376-5

[Stoica u. a. 2001] Stoica, Ion ; Morris, Robert ; Karger, David ; Kaashoek, Frans ; Balakrishnan, Hari: Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In: *Proceedings of the 2001 ACM SIGCOMM Conference*, 2001, S. 149–160

[Tanenbaum 2000] Tanenbaum, Andrew S.: *Computernetzwerke*. 3. revidierte Auflage. München : Pearson Studium, 2000

[U.S. Dept. Commerce/NIST und National Technical Information Service 1995] U.S. Dept. Commerce/NIST ; National Technical Information Service: *Secure Hash Standard*. 1995

[Yatin Chawathe and Sylvia Ratnasamy and Lee Breslau and Nick Lanham and Scott Shenker 2003] Yatin Chawathe and Sylvia Ratnasamy and Lee Breslau and Nick Lanham and Scott Shenker: Making gnutella-like P2P systems scalable. In: *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA : ACM, 2003, S. 407–418. – ISBN 1-58113-735-4

## Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

Unterschrift: _____

Stuttgart, 2008–08–04