LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

INSTITUT FÜR INFORMATIK
LEHRSTUHL FÜR MOBILE UND VERTEILTE SYSTEME

MASTERARBEIT

Organisation von Bloom-Filtern zur effizienten k-nächste-Nachbarn-Suche in kontextzentrischen sozialen Netzen

Judith Greif

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
LEHRSTUHL FÜR MOBILE UND VERTEILTE SYSTEME

MASTERARBEIT

Organisation von Bloom-Filtern zur effizienten k-nächste-Nachbarn-Suche in kontextzentrischen sozialen Netzen

Judith Greif

Aufgabenstellerin:	Prof. Dr. Claudia Linnhoff-Popien
Betreuer:	Mirco Schönfeld Dr. Martin Werner
Abgabetermin:	26. Juli 2016

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 26. Juli 2016

. . . . . .
(Unterschrift der Kandidatin)

Abstract

Hier kommt der Abstract hin. Hier kommt der Abstract hin.

Inhaltsverzeichnis

1 Einleitung

Die digitale Kommunikation hat im letzten Jahrzehnt einen rapiden Wandel erlebt. Soziale Online-Netzwerke[1] haben an Bedeutung gewonnen und zu neuen Kommunikationsmustern im Internet geführt. Sofortnachrichtendienste und Instant-Messenger ersetzen zunehmend Kommunikationsformen wie SMS und Telefonie. Das Smartphone hat das Mobiltelefon als mobiles Endgerät fast vollständig abgelöst. Feststehende Desktop-Rechner mit einem gleich bleibenden Netzzugang sind außerhalb von Firmen und Bildungseinrichtungen rückläufig. Dagegen sind Notebooks und Tablets weiterhin auf Erfolgszug[2].

Die Kommunikation im Internet ist jedoch nach wie vor Ende-zu-Ende- beziehungsweise adressbasiert. Das spiegelt sich im Aufbau der bestehenden sozialen Online-Netzwerke wider: Kommunikation basiert darin auf Online-Freundschaft. Mobilität und spezifischer Kontext der Mitglieder werden kaum berücksichtigt. In der Realität verlieren die Webbrowser-Schnittstellen der sozialen Netzwerke jedoch an Bedeutung. So veröffentlichte Facebook 2014 eine Studie zum Nutzerverhalten von US-Bürgern in einer Multigeräte-Welt[3]. Danach nutzten 60% der Erwachsenen in den USA täglich mindestens zwei Endgeräte, knapp 25% sogar drei Geräte. Mehr als 40% begannen den Tag mit einem Gerät und beendeten ihn mit einem anderen. Das Smartphone ist dabei das Gerät, das am häufigsten mitgenommen wird und eine zentrale Rolle in der Kommunikation per E-Mail und in sozialen Netzen einnimmt. Das Szenario, in dem Alice vor ihrem Desktop-Rechner zu Hause oder im Büro sitzt und über die Webbrowser-Schnittstelle Nachrichten an ihren Facebook-Freund Bob schreibt, gehört demnach der Vergangenheit an. Stattdessen verwendet Alice wohl eher ein Tablet, ein Notebook und ein Smartphone und kommuniziert mit Bob je nach Aufenthaltsort und Kontext ganz unterschiedlich.

So stellt sich die Frage nach einer Neuorientierung der sozialen Online-Netze: Nicht nur in der praktischen Umsetzung, also durch Schaffung unterschiedlicher Schnittstellen und neuer Funktionalitäten, sondern im Sinne eines tatsächlichen Paradigmenwechsels. Ein *kontextzentrisches soziales Netz*[4] basiert in seiner Struktur nicht auf Ende-zu-Ende-Kommunikation, adressbasiertem Routing und einem gleich bleibenden Netzzugang. Kommunikation beruht allein auf Kontext-Ähnlichkeit statt auf virtueller Freundschaft. Diese Überlegungen sind z.B. in das soziale Online-Netz AMBIENCE eingeflossen[5]. Nachrichten werden darin auf Grund von zeitlicher und räumlicher Ähnlichkeit ausgetauscht, Sender und Empfänger bleiben weitgehend anonym. Ein solches Netz erfordert eine neue Kommunikationsstruktur. Nachrichten werden nicht aktiv von einem Sender für

[1] Der englische Begriff hierfür lautet *Online Social Network (OSN)*. Die Begriffe *Soziales Netz*, *Soziales Online-Netz* und *Soziales Online-Netzwerk* werden im Folgenden synonym verwendet.

[2] Laut Analysen der Marktforschungsinstitute Gartner und IDC, vgl. z.B. `http://www.golem.de/news/pc-markt-absatz-von-pcs-geht-weiter-erheblich-zurueck-1601-118505.html`.

[3] Vgl. `https://www.facebook.com/business/news/Finding-simplicity-in-a-multi-device-world`.

[4] Der Begriff *Context awareness* wurde von Schilit et al. 1994 eingeführt und bezeichnet die Nutzung von Kontextinformationen als Informationsquelle für Anwendungen und Netzwerke. Hier ist vor allem *Context awareness* bezüglich des Nutzers von Interesse (vgl. [WDS15] für eine exakte Abgrenzung).

[5] Vgl. ebd. sowie Kap. 2 für eine detaillierte Darstellung.

einen spezifischen Empfänger verfasst und an ihn verschickt. Stattdessen kann ein Sender eine Nachricht verfassen und z.B. an einem WiFi-Access Point hinterlegen. Mitglieder des Netzwerks, die sich in der Nähe des Access Points aufhalten, können die dort vorhandenen Nachrichten mit gezielten Anfragen durchsuchen und die zum jeweiligen Kontext ähnlichsten Nachrichten abrufen.

Damit stellt sich die Frage: Wie lassen sich die Nachrichten an einem Host, also z.B. an einem WiFi-Access Point, so organisieren, dass die k ähnlichsten Nachrichten möglichst schnell und effizient gefunden werden? Wenn das soziale Netz wachsen und über den Status eines Prototypen hinaus erfolgreich sein soll, ist das von entscheidender Bedeutung. Mengentheoretisch betrachtet handelt es sich dabei um das Problem der k-nächsten-Nachbarn-Suche, die zu einer Anfrage die k ähnlichsten Elemente einer Menge, hier bestehend aus den Nachrichten an einem Host, finden soll.

Damit verknüpft ist die Frage nach der Nachrichtenform. Wie können Multimedia-Nachrichten wie Bilder, Textdateien oder Links effizient, einheitlich und sicher vor unbefugtem Zugriff hinterlegt und verschickt werden? Zudem muss die Ähnlichkeit oder Unähnlichkeit von Nachrichten ermittelt werden können, d.h. der k-nächste-Nachbarn-Suche muss ein Ähnlichkeitsmaß zu Grunde liegen. AMBIENCE verwendet dazu eine Bloom-Filter-Konstruktion. Eine Nachricht wird als Menge von Zeichenketten aufgefasst, die mit geeigneten Hashfunktionen in ein Bit-Array fester Länge eingefügt werden. Ähnlichkeit zwischen Nachrichten ist damit als Ähnlichkeit zwischen Bloom-Filtern definiert. Nachrichten werden in Form von Bloom-Filtern kodiert, gespeichert und verglichen. Als Ähnlichkeitsmaß dient die Jaccard-Distanz, mit der sich die Ähnlichkeit von Mengen beschreiben lässt.

Die folgende Arbeit behandelt daher die Organisation von Bloom-Filtern zur effizienten k-nächste-Nachbarn-Suche in kontextzentrischen sozialen Netzen. Das folgende Kapitel 2 gibt einen Überblick über mengentheoretische und probabilistische Grundlagen, verwendete Datenstrukturen und ihre Nutzung in AMBIENCE. Kapitel 3 gibt einen Überblick über verwandte Arbeiten und Fragestellungen. Das entwickelte Verfahren wird anschließend in Kapitel 4 dargestellt. Kapitel 5 vergleicht die Implementierung mit dem bisherigen, nicht optimierten Ansatz. Im abschließenden Kapitel 6 wird ein Fazit gezogen und auf mögliche zukünftige Arbeiten eingegangen.

2 Hintergrund

Im Folgenden werden grundlegende Begriffe, mathematische und probabilistische Verfahren sowie Daten- und Indexstrukturen dargestellt, die Eingang in diese Arbeit gefunden haben. Es wird erläutert, inwiefern sie für das soziale Netzwerk AMBIENCE relevant sind oder darin verwendet werden.

2.1 Bloom-Filter

Ein Bloom-Filter ist eine probabilistische Datenstruktur zur Beschreibung von Mengen, die in der ursprünglichen From 1970 von Burton H. Bloom eingeführt wurde[1]. Er besteht aus einem Bit-Array der festen Länge m, dessen Elemente zunächst alle auf 0 gesetzt sind. Das Einfügen von Informationsobjekten basiert auf der Berechnung einer festen Anzahl k unabhängiger Hashfunktionen, die positive Werte kleiner als m annehmen. Soll ein Objekt in den Filter eingefügt werden, werden seine Hashwerte berechnet und die die entsprechenden Bits im Filter gesetzt[2].

Die Hashwerte werden verwendet, um Anfragen auf dem Filter auszuführen. Ist ein Objekt im Filter enthalten, sind seine Bits gesetzt worden, d.h. man kann eine Anfrage nach seinen Hashwerten durchführen und so mit großer Wahrscheinlichkeit ermitteln, ob es in den Filter eingefügt wurde: Sind ein oder mehrere Bits des Anfrageobjekts nicht gesetzt, so ist das Element mit Sicherheit nicht vorhanden. Es gibt also keine falsch negativen Antworten. Allerdings kann es sein, dass ein Element nicht in den Filter eingefügt wurde, obwohl alle seine Bits gesetzt sind (falsch positive Antworten). Grund dafür ist die Kollisionseigenschaft von Hashfunktionen, die ein großes Universum von Elementen auf einen sehr viel kleineren Wertebereich, hier die Länge des Bloom-Filters, abbilden. Es kann somit zu Kollisionen zwischen unterschiedlichen Informationsobjekten bzw. ihren charakteristischen Hashwerten kommen. Die Falsch-Positiv-Rate eines Bloom-Filters ist abhängig von m, k und der Anzahl der eingefügten Elemente n und lässt sich berechnen als[3]

$$f = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k.$$

Aus der Kollisionseigenschaft folgt auch, dass ein einmal eingefügtes Objekt nicht mehr aus einem Bloom-Filter gelöscht werden kann, weil das offensichtlich zu falsch negativen Ergebnissen für andere Objekte führen könnte.

[1] Vgl. [Blo70].
[2] Vgl. [BM04]: 487.
[3] Vgl. ebd. für eine umfassende Darstellung.

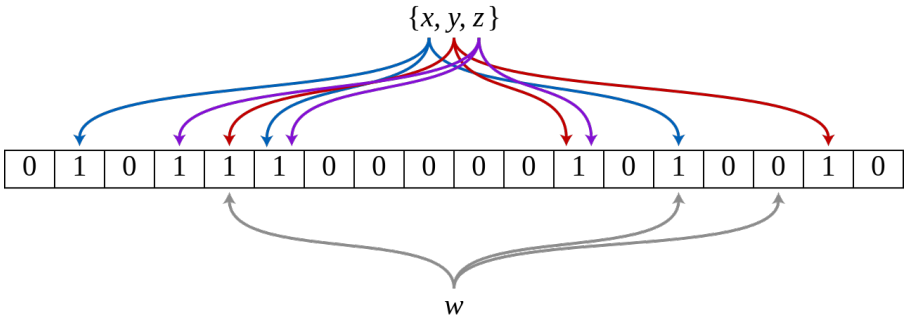

Abbildung 2.1: Beispiel für einen Bloomfilter, in den die Objekte x, y und z eingefügt wurden. Das Objekt w ist nicht im Filter vorhanden.

2.1.1 Distanzmaße

Um die Ähnlichkeit zweier Mengen zu ermitteln, werden verschiedene Distanzmetriken oder Ähnlichkeitsmaße eingesetzt. Um Bloom-Filter miteinander zu vergleichen und insbesondere für die k-nächste-Nachbarn-Suche muss ein geeignetes Ähnlichkeitsmaß zur Anwendung kommen. Bayardo et al. verwenden dazu die *Kosinus-Ähnlichkeit*[4]. Sakuma und Sato definieren die Ähnlichkeit von Bloom-Filtern über die Anzahl gleicher 1-Bits[5]. Ist diese für zwei Anfragefilter identisch, werden die Bit-Arrays negiert und die Anzahl gleicher 0-Bits verglichen.

AMBIENCE verwendet eine Abschätzung der *Jaccard-Distanz* zur Ermittlung der Ähnlichkeit von Bloom-Filtern. Die Jaccard-Distanz zwischen zwei Mengen A und B ist definiert als

$$J_\delta(A,B) = 1 - J(A,B) = \frac{|A \cap B| - |A \cup B|}{|A \cup B|},$$

wobei

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

den *Jaccard-Koeffizienten* bezeichnet. Die Jaccard-Distanz nimmt Werte im Bereich $[0,1]$ an. Identische Mengen haben eine Jaccard-Distanz von 0, Mengen ohne gemeinsame Elemente haben eine Jaccard-Distanz von 1. Für Bloom-Filter lässt sich die Jaccard-Distanz analog berechnen. Die Vereinigungsmenge zweier Bloom-Filter F und G lässt sich als bitweises logisches Oder, die Schnittmenge als bitweises logisches Und repräsentieren. Auch hier nimmt die Jaccard-Distanz offensichtlich Werte zwischen 0 und 1 an. Je ähnlicher die Filter sind, desto kleiner ist ihre Jaccard-Distanz. Die Jaccard-Distanz zwischen Bloom-Filtern ist, anders als die von Sakuma und Sato verwendete Distanz-Metrik, nicht transitiv in dem Sinne, dass zwei Filter, die beide eine geringe Jaccard-Distanz zu einem dritten Filter aufweisen, untereinander nicht ähnlich sein müssen. Man betrachte z.B. die Filter *F1*, *F2* und *F3* mit folgenden Werten:

[4] Vgl. [BMS07].
[5] Vgl. [SS11]: 321.

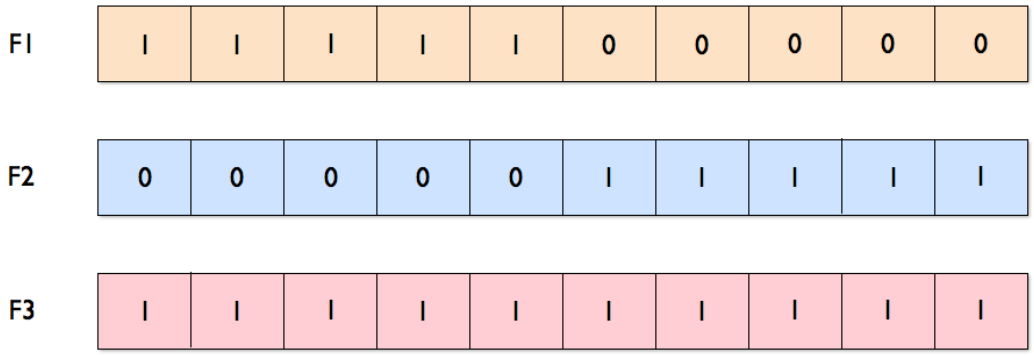

Abbildung 2.2: Jaccard-Distanzen zwischen Bloom-Filtern.

Die Jaccard-Distanzen zwischen *F1*, *F2* und *F3* betragen somit:

$$J_\delta(F1, F3) = 1 - J(F1, F3) = 1 - \frac{|F1 \cap F3|}{|F1 \cup F3|} = 1 - \frac{5}{10} = 0.5$$

$$J_\delta(F2, F3) = 1 - J(F2, F3) = 1 - \frac{|F2 \cap F3|}{|F2 \cup F3|} = 1 - \frac{5}{10} = 0.5$$

$$J_\delta(F1, F2) = 1 - J(F1, F2) = 1 - \frac{|F1 \cap F2|}{|F1 \cup F2|} = 1 - 0 = 1$$

Daran wird deutlich: Obwohl *F1* und *F2* jeweils die Hälfte der Elemente mit *F3* gemeinsam haben, lässt sich daraus kein Wert für die Ähnlichkeit zwischen *F1* und *F2* ableiten. Sie sind sich im Gegenteil maximal unähnlich.

2.1.2 Teil- und Obermengenbeziehung

Will man Bloom-Filter z.B. nach Ähnlichkeit gruppieren, kann man stattdessen Teil- und Obermengenbeziehungen zwischen ihnen betrachten. Die *Teilmengenbeziehung* zwischen zwei Bloom-Filtern sei hier wie folgt definiert:

> Ein Bloom-Filter F ist ***Teilmenge*** eines Bloom-Filters G, wenn darin mindestens die gleichen 0-Bits gesetzt sind wie in G (und möglicherweise weitere, zusätzliche 0-Bits).

Die Obermengenbeziehung zwischen zwei Bloom-Filtern sei hier wie folgt definiert:

> Ein Bloom-Filter F ist ***Obermenge*** eines Bloom-Filters G, wenn darin mindestens die gleichen 1-Bits gesetzt sind wie in G (und möglicherweise weitere, zusätzliche 1-Bits).

Der maximal gefüllte Filter, in dem alle Bits gesetzt sind, ist damit Obermenge aller Filter derselben Länge (auch von sich selbst). Der leere Filter ist die (triviale) Teilmenge aller Filter derselben Länge (auch von sich selbst). Teil- und Obermengen sind Umkehrungen

voneinander, d.h. wenn F Teilmenge von G ist, folgt daraus, dass G Obermenge von F ist, und umgekehrt.

Zwischen den Bloom-Filter aus Abb. 2.2 bestehen folgende Teil- und Obermengenbeziehungen: *F1* und *F2* sind Teilmengen von *F3*. *F3* ist Obermenge von *F1* und *F2*. Zwischen den maximal unähnlichen Filtern *F1* und *F2* bestehen keine Teil- und Obermengenbeziehungen. Teil- und Obermengenbeziehung sind außerdem transitiv, was an

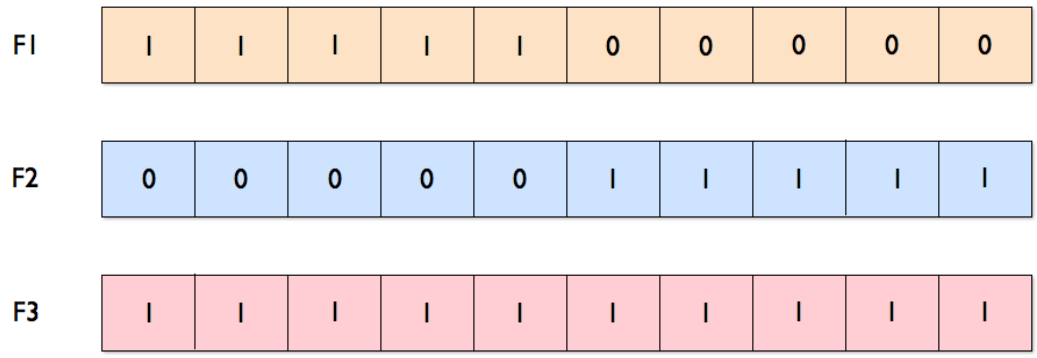

Abbildung 2.3: Teil- und Obermengenbeziehungen zwischen Bloom-Filtern.

Abb. 2.3 deutlich wird: *F1* ist Teilmenge von *F2*, damit auch Teilmenge von *F3*. *F3* ist Obermenge von *F2*, damit auch Obermenge von *F1*.

Teil- und Obermengenbeziehungen sind also im Gegensatz zur Jaccard-Distanz dazu geeignet, transitive Ähnlichkeitsbeziehungen zwischen Bloom-Filtern abzubilden. Diese Eigenschaft spielt eine zentrale Rolle im hier entwickelten Verfahren, das in Kap. 4 ausführlich dargestellt wird.

2.1.3 Hashfunktionen

Die Frage, welche Hashfunktionen optimalerweise für einen Bloom-Filter verwendet werden sollten, ist nicht eindeutig zu beantworten[6]. Grundsätzlich muss zwischen kryptografischen Hashfunktionen wie MD5 und SHA und gewöhnlichen Hashfunktionen wie Murmur- oder Jenkins-Hashfunktionen unterschieden werden. Die Berechnung von kryptografischen Hashfunktionen dauert in der Regel länger, dafür haben sie bestimmte Eigenschaften wie eine hohe Kollisionsresistenz und Gleichverteilung der Ergebniswerte.

Werden Bloom-Filter z.B. zum schnellen Nachschlagen in großen, verteilten Datenbanken eingesetzt, wird auf die kryptografischen Eigenschaften zu Gunsten des verminderten Rechenaufwandes verzichtet. Das NoSQL-Datenbanksystem Cassandra und das Hadoop-Framework für skalierte, verteilt arbeitende Software verwenden beispielsweise Bloom-Filter in Kombination mit Murmur- und Jenkins-Hashfunktionen. Darüber hinaus ist MD5 für Bloom-Filter weit verbreitet. Die Murmur-Hashfunktionen weisen gleichzeitig gute Verteilungseigenschaften auf und lassen sich vergleichsweise schnell berechnen, weswegen sie generell für den Einsatz in Bloom-Filtern empfohlen werden[7]. AMBIENCE

[6] Vgl. [BM04]: 487.

[7] Vgl. `http://spyced.blogspot.de/2009/01/all-you-ever-wanted-to-know-about.html`.

verwendet Murmur-Hashfunktionen zur Generierung der Bloom-Filter. Für die eigene Implementierung wurde der Murmur2-Hash verwendet[8].

2.1.4 Varianten und Anwendungen

Bloom-Filter erfreuen sich in verschiedenen Versionen großer Beliebtheit. Wichtige Varianten sind z.B. *Attenuated Bloom Filter*[9], *Counting Bloom-Filter* und *Compressed Bloom Filter*. Ein Counting Bloom-Filter benötigt mehr Speicherplatz als ein klassischer Bloom-Filter, dafür können Objekte wieder daraus entfernt werden[10]. Komprimierte Bloom-Filter werden eingesetzt, wenn Bloom-Filter als Nachrichten versendet werden und die Nachrichtenlänge begrenzt ist oder die übertragene Datenmenge minimiert werden soll[11]. Attenuated Bloom-Filter können als Array von Bloom-Filtern betrachtet werden und können z.B. in einem Netzwerk Informationen darüber enthalten, welche Dienste an einem anderen Knoten im Netzwerk verfügbar sind. Wegen der guten Kompressionseigenschaften und des geringen Speicherbedarfs kommen Bloom-Filter häufig in verteilten Anwendungen und Netzwerkdiensten zum Einsatz[12].
Neben Hadoop und Cassandra werden Bloom-Filter in unzähligen, zum Teil hoch skalierenden Anwendungen eingesetzt. Weitere Beispiele sind der quelloffene Webproxy Squid und der Chrome-Browser, wo Bloom-Filter zum schnellen Nachschlagen von als bösartig eingestuften Webseiten verwendet werden. Broder und Mitzenmacher fassen das Bloom-Filter-Prinzip wie folgt zusammen:

> Wherever a list or set is used, and space is at a premium, consider using a Bloom filter if the effect of false positives can be mitigated[13].

2.2 Indexstrukturen

To support query processing and operations in an efficient manner, the internal layer of a database system uses specific data strucures and memory methods. These are called *index structures*. They organize the data to support the required operations using its *indices*.

An *index* (also called *directory*) of a file holds information about its structure. A *file* in this context refers to an entire data structure, i.e. an array, a search tree etc.. One can differentiate between three classes of index structures depending on the manner of organization:

1. ***Data-organizing index structures*** are used to organize the actual amount of data. They mostly rely on *search trees*.
2. ***Space-organizing index structures*** are used to organize the space that holds the data. They make use of *dynamic hashing*.
3. ***Hybrid index structures*** are a combination of both classes. They are based on *hash trees*.

There are several requirements for an index structure in order to meet its purpose.

[8] Vgl. `https://sites.google.com/site/murmurhash/MurmurHash2.cpp` für den Quellcode.
[9] Vgl. [SS11]: 316 und 318.
[10] Vgl. [FCAB00].
[11] Vgl. [Mit02].
[12] Vgl. [BM04] für eine ausführliche Darstellung.
[13] Vgl. [BM04]: 486.

- *Efficient search:* A data query on the index structure should return an answer in optimal time, i.e. the query should be directed to the page or pages that contain the queried data using as little steps as possible.
- *Dynamic insertion, deletion and modification of data sets:* The amount of data to be organized changes over time, leading to alterations in the index structure as well. Any implementation requiring a complete reorganization of the index structure on insertion, deletion or modification of data sets is clearly unacceptable. Any of these operations may therefore only lead to local changes.
- *Local preservation of order:* If there are some data sets the keys of which are successors within the applied order relation (i.e. the less-or-equal relation on non-negative integers), this order should be preserved within the index structure. This holds for search trees but it does not hold for linear hashing. It is clearly of great importance regarding the application scenario in question.
- *Efficient use of space:* This requirement is of great importance for real-world applications. So far the reference implementation *AMBIENCE* has served as a proof of concept. Accordingly the number of messages, i.e. the amount of data to be queried, has been relatively small compared to a real-world scenario. Therefore the memory requirements of any index structure within the current scenario that represents the actual amount of data is unlikely to require vast amounts of memory. However, keeping in mind future application scenarios for *AMBIENCE*, efficient use of space cannot be entirely discarded.

Further requirements include *feasability* and *implementation cost*. Any index structure aiming at a real-world implementation such as *AMBIENCE* naturally has to be feasable, so this requirement will be overlooked in the following. As this work clearly has a scholarly background, not an industrial one, the implementation cost will be disregarded as well. [OW12]

2.2.1 B- und B+-Bäume

[Knu98]

2.3 AMBIENCE

Referenzen bis jetzt: [AT06], [ADI+12], [BMS07], [BCM02], [DWM10], [HP94], [LC86], [Naf05], [QLC14], [RK14], [SBE+12], [Sch13], [SW14], [STT+09], [YL02], [Zha12], [ZJW04], [Jan95].

3 Verwandte Themen

4 Implementierung

5 Evaluation

6 Zusammenfassung und Ausblick

Abbildungsverzeichnis

Literaturverzeichnis

[ADI+12] Ahlgren, Bengt, Christian Dannewitz, Claudio Imbrenda, Dirk Kutscher und Börje Ohlman: *A Survey of Information-Centric Networking*. Communications Magazine, IEEE, 50(7):26–36, 2012.

[AT06] Agarwal, Sachin und Ari Trachtenberg: *Approximating the number of differences between remote sets*. In: *Information Theory Workshop, 2006. ITW '06 Punta del Este. IEEE*, Seiten 217–221, March 2006.

[BCM02] Byers, John, Jeffrey Considine und Michael Mitzenmacher: *Fast Approximate Reconciliation of Set Differences*. In: *BU Computer Science TR*, Seiten 2002–2019, 2002.

[Blo70] Bloom, Burton: *Space/Time Trade-offs in Hash Coding with Allowable Errors*. Communications of the ACM, 13(7):422–426, 1970.

[BM04] Broder, Andrei und Michael Mitzenmacher: *Network Applications of Bloom Filters: A Survey*. Internet Mathematics, 1(4):485–509, 2004.

[BMS07] Bayardo, Roberto, Yiming Ma und Ramakrishnan Srikant: *Scaling Up All Pairs Similarity Search*. In: *Proceedings of the 16th international conference on World Wide Web*, Seiten 131–140. ACM, 2007.

[DWM10] Dürr, Michael, Martin Werner und Marco Maier: *Re-Socializing Online Social Networks*. In: *Green Computing and Communications (GreenCom), 2010 IEEE/ACM International Conference on & International Conference on Cyber, Physical and Social Computing (CPSCom)*, Seiten 786–791. IEEE, 2010.

[FCAB00] Fan, Li, Pei Cao, Jussara Almeida und Andrei Broder: *Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol*. IEEE/ACM Transactions on Networking (TON), 8(3):281–293, 2000.

[HP94] Hellerstein, Joseph und Avi Pfeffer: *The RD-Tree: An Index Structure for Sets*. Technischer Bericht, University of Wisconsin-Madison, Computer Sciences Department, 1994.

[Jan95] Jannink, Jan: *Implementing Deletion in B+-Trees*. ACM Sigmod Record, 24(1):33–38, 1995.

[Knu98] Knuth, Donald: *The art of computer programming, Volume 3, Sorting and searching*. Addison Wesley Longman, 1998.

[LC86] Lehman, Tobin und Michael Carey: *A Study of Index Structures for Main Memory Database Management Systems*. In: *Proc. VLDB*, 1986.

[Mit02] Mitzenmacher, Michael: *Compressed Bloom Filters*. IIEEE/ACM Transactions on Networking (TON), 10(5):604–612, 2002.

[Naf05] Nafe, Clemens: *Indexierung lokaler Daten in Peer-to-Peer-Netzwerken.* Diplomarbeit, Universität Rostock, 2005.

[OW12] Ottmann, Thomas und Peter Widmayer: *Algorithmen und Datenstrukturen.* Spektrum Akademischer Verlag, 5 Auflage, 2012.

[QLC14] Qiao, Yan, Tao Li und Shigang Chen: *Fast Bloom Filters and their Generalization.* Parallel and Distributed Systems, IEEE Transactions on, 25(1):93–103, Januar 2014.

[RK14] Ruppel, Peter und Axel Küpper: *Geocookie: A Space-Efficient Representation of Geographic Location Sets.* Journal of Information Processing, 22(3):418–424, 2014.

[SBE+12] Sarwat, Mohamed, Jie Bao, Ahmed Eldawy, Justin Levandoski, Amr Magdy und Mohamed Mokbel: *Sindbad: A Location-(B)ased Social Networking System.* In: *Proceedings or the 2012 ACM SIGMOD International Conference on Management of Data*, Seiten 649–652. ACM, 2012.

[Sch13] Schnell, Rainer: *Getting Big Data But Avoiding Big Brother.* WP-GRLC, 2, 2013.

[SS11] Sakuma, Hiroshi und Fumiako Sato: *Evaluation of the Structured Bloom Filters Based on Similarity.* In: *Advanced Information Networking and Applications (AINA), 2011 IEEE International Conference on*, Seiten 316–323, März 2011.

[STT+09] Shiraki, Toru, Yuichi Teranishi, Susumu Takeuchi, Kaname Harumoto und Shojiro Nishio: *A Bloom Filter-Based User Search Method Based on Movement Records for P2P Network.* In: *Applications and the Internet, 2009. SAINT '09. Ninth International Symposium on*, Seiten 177–180. IEEE, Juli 2009.

[SW14] Schönfeld, Mirco und Martin Werner: *Node Wake-(U)p via OVSF-Coded Bloom Filters in Wireless Sensor Networks.* In: *Ad Hoc Networks*, Seiten 119–134. Springer, 2014.

[WDS15] Werner, Martin, Florian Dorfmeister und Mirco Schönfeld: *AMBIENCE: A Context-Centric Online Social Network.* In: *12th IEEE Workshop on Positioning, Navigation and Communications (WPNC '15)*, 2015.

[YL02] Yang, Congjun und King-Ip Lin: *An Index Structure for Improving Closest Pairs and Related Join Queries in Spatial Databases.* In: *Database Engineering and Applications Symposium, 2002. Proceedings. International*, Seiten 140–149. IEEE, 2002.

[Zha12] Zhang, Zhenghao: *Analog Bloom Filter: Efficient simultaneous query for wireless networks.* In: *Global Communications Conference (GLOBECOM), 2012 IEEE*, Seiten 3340–3346. IEEE, 2012.

[ZJW04] Zhu, Yifeng, Hong Jiang und Jun Wang: *Hierarchical Bloom Filter Arrays (HBA): A Novel, Scalable Metadata Management System for Large Cluster-based Storage.* In: *Cluster Computing, 2004 IEEE International Conference on*, Seiten 165–174. IEEE, 2004.