Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung - Anwendungen 2

Julissa Cusi Juarez

Skyline Query Processing

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Julissa Cusi Juarez

Skyline Query Processing

Ausarbeitung eingereicht im Rahmen der Vorlesung Anwendungen II
im Studiengang Informatik Master of Science
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuer: Prof. Dr. Olaf Zukunft

Abgegeben am 31.08.2010

Inhalt

1. Einleitung

In der heutigen Zeit ermöglicht die hohe Rechenleistung von PC´s dass diese an verteilten Systemen, wie P2P Systemen, teilnehmen können. Der Bedarf Informationen zu teilen führt zur Implementierung von P2P Datenbanken Systemen. P2P Overlays, die Datenbanken-Funktionalitäten unterstützen, sind eine populäre Auswahl um große verteilte Data-Management-Systeme aufzubauen [2].

In einem P2P System, während eine Query-Verarbeitung realisiert wird, können Knoten ausfallen, das Netz verlassen, keine Antwort geben oder die Antwort verzögern. Daher sind Knoten und die dort gespeicherten Daten nicht permanent verfügbar. Aber in einer bestimmten Zeit kann man eine große Menge Knoten bzw. Informationen finden. Dieses Szenario ist ideal für statistische Anfragen wie Top-k Queries, wobei Aggregation und Rankings genutzt werden kann. Ein Typ von statistischen Anfragen sind Skyline Queries, welche für multikriterielle Entscheidungsunterstützung benutzt werden.

Ein Skyline Query liefert als Anfrageergebnisse eine Menge von Datenobjekten aus der Datenbank zurück. Das Query Dataset wird in einem n-dimensionalen Raum repräsentiert und jede Dimension entspricht einem Query Attribut. Ein Skyline Query Beispiel wäre folgendes:

„Die preisgünstigsten Hotels, die sich am nächsten zum Flughafen befinden“

Die Query hat zwei Dimensionen: Preis und Abstand zum Flughafen. Die besten Daten, die beide Bedingungen erfüllen, sind in einem zweidimensionalen Raum durch Punkte repräsentiert. Diese Punkte bilden eine Linie bzw. eine Skyline.

Skyline Queries sind für Top-k Queries Verarbeitung geeignet und können sich gut in einer dynamischen Umgebung, wie P2P Netzen, anpassen. In dieser Arbeit werden verschiedene Skyline Query VerarbeitungsMethoden präsentiert.

1.1. Aktuelle Arbeit

Derzeit gibt es keine Standard Methode um Skyline Queries zu verarbeiten, stattdessen gibt es viele Forschungsarbeiten.

Die ersten Algorithmen um Skyline Queries zu verarbeiten wurden im Jahre 2001 präsentiert. Die Algorithmen Block-Nested-Loops (BNL) und Divide-and-Conquer (D&C) sind die Basis für Skyline Query Forschungsarbeiten, wenn sie Indexstrukturen wie R-Tree und B-Tree verwenden.

Demnach werden die genannten Algorithmen mittels einer Sortierung der Input Daten optimiert. Der nächste Schritt ist die progressive Verarbeitung. In diesem Fall werden die Input Daten oder das Dataset in einen Dimensionalen Raum umgewandelt. Ein anderer Ansatz ist die Datenpunkte im Bezug auf einen Query Punkt zu klassifizieren, dass heißt Datenpunkte zu finden, die am nächsten zu einem Query Punkt stehen, wie der Algorithmus Nearest Neighbors (NN).

Weitere Forschungen arbeiten in den Bereichen der Teilung des Dimensionalen Raumes oder in Subspaces in der Steuerung von Skylines, besonders in im Dataset Aktualisierungs Fall, in der Skyline Query Verarbeitung über Data Streams, oder in Subspaces mit Einschränkungen oder auch in einem hohen dimensionalen Raum.

Es gibt auch Forschungsarbeiten in komplexeren Data-Umgebungen wie top-k Skyline mit der maximalen Anzahl der dominierenden Datenpunkte, Spatial Skyline, unter Berücksichtigung einer Gruppe von Query Punkten als Referenz bzw. Nachbarschaft unter anderen komplexen Dataumgebung.

Die zentralisierten Algorithmen lassen sich nicht direkt auf verteilte Umgebungen wie P2P Netze anwenden. Allerdings gibt es verschiedene Forschungsarbeiten die in verteilten Umgebungen arbeiten, einige in small-

scale und andere in large-scale verteilten Umgebungen. Letztere versuchen in Subspaces zu arbeiten und lokale Skyline Berechnung zu realisieren, um die unnötige Datenübertragung zwischen Peers zu reduzieren, sowie eine parallele Skyline Suche durchzuführen.

2. Personifizierte Top-k Skyline Queries

Die große verfügbare Datenmenge im Web erhöht die Wichtigkeit intelligenter Anfragemechanismen zu implementieren, damit die Benutzer die idealen Anfrageergebnisse erhalten. Ein Mechanismus dazu ist, dank der intuitiven Anfrageformulierung,Skyline Query Processing[5].

Skyline Query Processing kann zu viele Ergebnisse ergeben, insbesondere wenn das Dataset eine hohe Raumdimensionalität hat, um den Bedarf einer großen Anzahl von Benutzern zu erfüllen. Dies kann ein Overhead Processing bewirken.

Das Ziel ist Personifizierte Top-k Skyline Queries zu unterstützen, dazu werden die idealen k-Anfrageergebnisse auf spezifischer Benutzerpräferenz basierend identifiziert. Die Top-k Ergebnisse werden durch die Navigation eines Skycubes identifiziert. Der Aufbau des Skycubes kann aber Storage- und Berechnungsoverhead bewirken, hauptsächlich wenn man eine hohe Dimensionalität und große Datenmenge hat.

Um den Overhead zu reduzieren benutzt der personifizierte Top-k Skyline Query Mechanismus vier verschiedene Ansätze, die in Folge präsentiert werden.

2.1. Subspaces Skyline

In einem Dataset mit n-Attributen oder Dimensionen, kann man Skyline Queries bezüglich aller Dimensionen haben oder einer Untermenge davon. Im Fall einer Untermenge der Dimensionen, spricht man von Subspaces [4].

Beispiel:

Ein Hotel-Dataset mit drei Attributen bzw. Dimensionen: Preis, Abstand zum Meer, Anzahl der Sterne. Wir betrachten folgende Queries:

1. **(3-dimensional)**
   Die preisgünstigsten Hotels, die sich am nächsten zum Meer befinden und am meisten Sterne haben.
2. **(2-dimensional)**
   Die preisgünstigsten Hotels, die sich am nächsten zum Meer befinden.
3. **(1-dimensional)**
   Die Hotels, die sich am nächsten zum Meer befinden.

Das erste Beispiel nutzt alle Dataset-Dimensionen. Die nachfolgenden nutzen nur eine Untermenge der Dimensionen.

Eine Untermenge wird ein Subspace sein, seine entsprechende Query ist eine Subspace Skyline Query und seine Skyline wird eine Subspace Skyline genannt.

2.2. Skycube

Ein Skycube besteht aus allen möglichen Subspaces Skylines von einem Dataset [3]. Mit dem vorherigen Beispiel.

Die Dimensionen:

Preis = A, Abstand zum Meer = B, Anzahl der Sterne = C

	A	B	C
t1	40	30	4
t2	50	10	5
t3	10	40	2
t4	30	50	1
t5	20	20	3

Tabelle1. Dataset (angelehnt an [4])

In Tabelle 1 wird das Beispiel-Dataset dargestellt. A, B und C sind die Dimensionen und t1 bis t5 sind die Tupel die die Anfrage am besten erfüllen.

Alle möglichen Kombinationen der verschiedenen Dimensionen sind Subspaces. Jeder Subspace wird im Folgenden „Cuboid" genannt.

Cuboid	Skyline
ABC	{t1, t2, t3, t5}
AB	{t2, t3, t5}
AC	{t1, t2, t3, t5}
BC	{t2}
A	{t3}
B	{t2}
C	{t2}

Tabelle 2. Cuboids (angelehnt an [4])

In Tabelle 2 werden alle möglichen Cuboids des Beispiel-Datasets dargestellt. Außerdem werden die Tupel die am besten die Subspace Query erfüllen in der Spalte Skyline präsentiert.

Ein Skycube kann in einer Lattice-Struktur visualisiert werden, sowie ein Datacube in einem Datawarehouse visualisiert werden kann. In diesen Fall wird jede Ebene von unten bis oben inkrementell nummeriert.

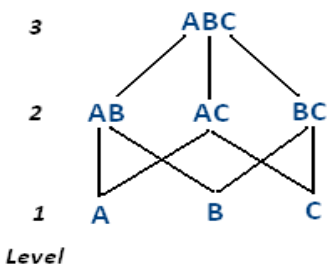

Fig. 1. Skycube - Lattice Struktur (angelehnt an [4])

2.3. Compressed Skycube (CSC)

In Tabelle 2 kann man sehen, dass ein Tupel oder Datenobjekt in mehrere Cuboids eines Skycubes gespeichert werden kann. Zum Beispiel ***t2***.

Falls das Dataset aktualisiert wurde, muss man den Cube neu berechnen. Eine neue Cube-Berechnung kann ein Query Processing Bottle-Neck bewirken [4].

Daher wurden Minimum Subspaces erfunden, womit Duplikate gelöscht werden können.
Ein Minimum Subspace ist ein Cuboid und behält nur die Datenobjekte, die nicht in seinen eigenen Subspaces definiert sind [4].

Tabelle 3 stellt die zusammengefasste Information von Tabelle 2 dar, ohne duplikate Datenobjekte. Zum Beispiel die beinhaltenden Tupel in Cuboid ABC sind gleichzeitig in ihren Subspaces gespeichert, folglich wird Cuboid ABC gelöscht.

Andererseits das Cuboid AB beinhaltet die Tupel t2, t3 y t5. Zwei davon, t2 und t3, sind jeweils in den Subspaces A und B gespeichert, daher wird das Cuboid AB nur die Tupel t5 behalten.

Cuboid	Skyline
AB	{t5}
AC	{t1, t3, t5}
A	{t3}
B	{t2}
C	{t2}

Tabelle 3. Cuboids (CSC) (angelehnt an [4])

Beim Umordnen der Tabelle 3 bezüglich der Tupel, wird es die Minimun Subspaces definieren. Die Minimum Subspaces sind die Cuboids in denen ein Tupel gespeichert ist.

	Minimum Subspaces
t1	{AC}
t2	{B, C}
t3	{AC}
t5	{AC}

Tabelle 4. Minimun Subspaces (angelehnt an [4])

Ein Compressed Skycube besteht aus Minimum Subspaces.

2.4. Framework „Microscope“

Das Framework „Microscope“ benutzt einen Compressed Skycube um das Dataset zu aktualisieren oder eine Suche um eine Anfrage zu beantwortet.

Die Verwendung eines Compressed Skycube ist vorteilhaft, weil wie geschildert die verschiedenen Benutzeranfragen über ein Dataset nicht alle Dimensionen anfragen. Andererseits kann die Aktualisierung eines Skyline-Cube Storage- und Processing-Overhead kosten.

Microscope navigiert in einem Compressed Skycube in einer Botton-up Weise, es fängt in den kleinsten Subspaces an und schreitet dann voran bis zu den größte. So kann man feststellen dass ein Space nur nach seinen eigenen Subspaces besucht wird [5].

Die Fig. 2 stellt eine Lattice-Struktur des CSC unseres Beispiels dar. In einem Navigationsfall erfolgt der Besuch der Subspaces in folgender Reihenfolge: A, B, AB, C, BC, AC, ABC.

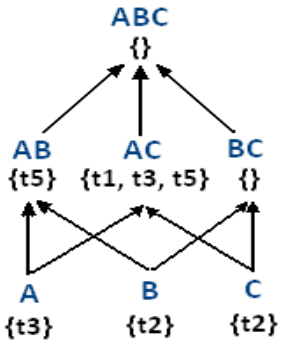

Fig. 2. Lattice Struktur des CSC (angelehnt an [4])

Der Microscope-Algorithmus wird mit dem Besuch jedes Cuboid die Skylines zu seiner eigenen Ergebnismenge hinzufügen.

2.5. Bewertung

Die Benutzung eines Compressed Skycube gibt für Microscope viele Vorteile, da sowohl das Storage- als auch das Processing-Overhead reduziert werden. Die Overhead-Ermäßigung geht aus der direkten Navigation des Compressed Skycubes hervor.

In einem Storage Overhead Test wurden Datenobjekte sowohl in einem normalen Skycube als auch in einem Compressed Skycube gespeichert. Der Test wurde mit unterschiedlichen Mengen von Dimensionen und Daten durchgeführt. In allen Fällen wurde bewiesen, dass ein Compressed Skycube effektiver ist und sein Storage Overhead über dem eines normalen Skycube liegt und mit zunehmender Anzahl der Dimensionen und Datenobjekte ansteigt.

Ebenso wurde die Effektivität des Microscope Framework gegenüber dem Naive Framework verglichen. Das Framework Naive benutzt auch ein Compressed Skycube, aber es hat keine Button-Up Navigationsweise. In diesem Test wurde bewiesen, dass die Kosten des Microscope leichter ansteigen als die des Framework Naive.

In einem P2P Netz können eine große Menge Information gefunden werden, daher kann man ein Dataset mit hoher Dimensionalität haben. Eine große Herausforderung in der Skyline Verarbeitung ist die Dimensionalität. Durch die Tests wurde bewiesen, dass Framework Microscope eine gute Leistung hat für Datasets mit hoher Dimensionalität. Aus diesem Grunde wird das Skyline Query Processing mit Framework Microscope als effizient und anwendbar für P2P Datenbanken bezeichnet.

3. Spatial Skyline in Wireless Sensor Networks

Spatial Skylines können in Wireless Sensor Networks für gemeinschaftliche Objekt-Positionierung benutzt werden.

Eine Anfrage wird durch einen Punkt in einem dimensionalen Raum gemacht, wobei eine Berechnung der gemeinschaftlichen Objekt-Positionierung in Bezug auf die Gruppe von Punkten bzw. Gruppenanfrage erfolgt [6].

Die Herausforderung ist das Design eines Algorithmus, der mit dimensionalen Daten arbeitet und ein effiziente Berechnung der Skylines realisieren kann.

Beispiel:

- Eine Gruppe Polizeiautos müssen verschiedene Polizeieinsätze bewältigen. Hierbei wird jeder Polizeieinsatz ein Query-Punkt sein und die Polizeiautos Daten-Punkte.
- Sowohl die Skyline Queries (Polizeieinsätze) als auch die Datenpunkte (Polizeiautos) werden jeweils durch einen Punkt in dem zweidimensionalen Raum repräsentiert. Dieser Punkt entspricht einer Position im Koordinatensystem.

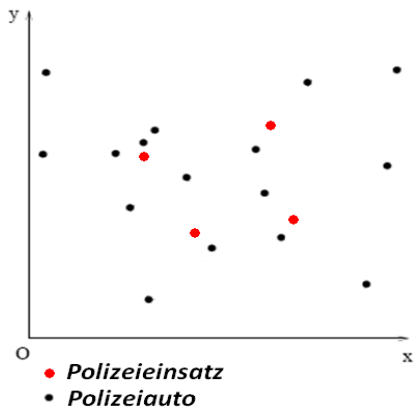

Fig. 3. Beispiel (Spatial Skyline)

Die Skyline lässt sich mit dem verteilten Algorithmus „Distributed Spatial Skyline (DSS)“ finden.

3.1. Distributed Spatial Skyline (DSS) Algorithmus

Das Ziel des DSS Algorithmus ist Spatial-Skylines in Wireless Sensor Networks zu finden. Dazu parallelisiert DSS die Skyline Suche mittels einer Partition des dimensionalen Raumes [6].

Die Suche wird in drei Phasen realisiert:

1. Partition des dimensionalen Raumes
2. Interne Skylines Suche
3. Externe Skylines Suche

1. DSS - Partition des dimensionalen Raumes

In dieser Phase werden folgende Begriffe benutzt:

- **Convex Hull**. Von mehrere Punkte in einen dimensionalen Raum. Die Convex Hull bzw. Konvexe Hülle wird das kleinste Polygon, das alle Punkte umfasst.

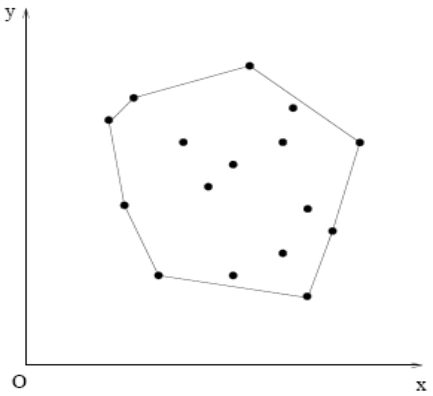

Fig. 4. Convex Hull [8]

- **Voronoi Diagramm**. Mit dem Voronoi-Diagramm wird eine Zerlegung des Raumes in Regionen bezeichnet, die durch eine vorgegebene Menge an Punkten des Raumes, hier als Zentren bezeichnet, bestimmt werden. Jede Region wird durch genau ein Zentrum bestimmt und umfasst alle Punkte des Raumes, die in Bezug zur euklidischen Metrik näher an dem Zentrum der Region liegen, als an jedem anderen Zentrum. Derartige Regionen werden auch als Voronoi-Regionen bezeichnet. Aus allen Punkten, die mehr als ein nächstgelegenes Zentrum besitzen und somit die Grenzen der Regionen bilden, entsteht das Voronoi-Diagramm [7].

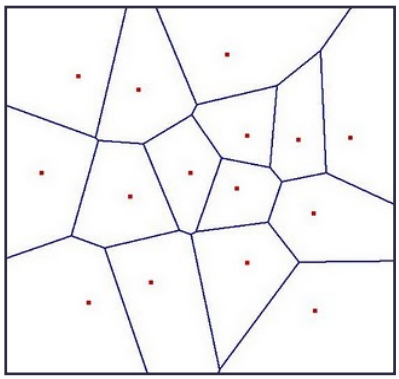

Fig.5. Voronoi Diagramm

In Dieser Phase partitioniert man den dimensionalen Raum bezüglich drei Weisen. Die Partitionierung wird in der Fig.6. dargestellt.

- **Convex Hull**

  Zuerst bestimmt man jeweils den Datenpunkt (0 bis 16 in Fig. 6.), der einem Query Punkt (A, B, C, D) am nächsten liegt. Diese Punkte werden t-Home (triangle-Home) Punkte genannt, Punkte 0, 14, 9, 2.

  Anschließend bestimmt man einen globalen Koordinator von allen Datenpunkten (Punkte 7). Er definiert eine Convex Hull in Bezug auf die Query Punkte und einen Schwerpunkt bzw. einen Rendezvous Punkt (Punkt R).

  Nach der Entstehung der Convex Hull (Polygon ABDC) sind alle Datenpunkte in 2 Bereiche unterteilt. Die Punkte, die innerhalb und außerhalb der Query Convex Hull sind.

- **Triangulation**

  Nach der Convex Hull Erstellung, gibt es zwei Bereiche. Man partitioniert beide Bereiche per Triangulation (Bereiche ARC, ARB, BRD, CRD). Der Algorithmus ermöglicht die parallele Suche per Dreieck.

  Die definierte t-Home Punkte in der vorherigen Partitionierung sind jeweils verantwortlich für ein Dreieck, das am nächsten im Uhrzeigersinn liegt. Dann in Fig. 6: 0 ist t-Home Punkt von ARC, 14 ist t-Home Punkt von BRD, 2 ist t-Home Punkt von DRC und 9 ist t-Home Punkt von ARC.

- **Voronoi**
  Der Algorithmus definiert Voronoi Bereiche in den ganzen Raum. Wobei jeder Datenpunkt (0 bis 16) das Zentrum einer Voronoi-Region ist.

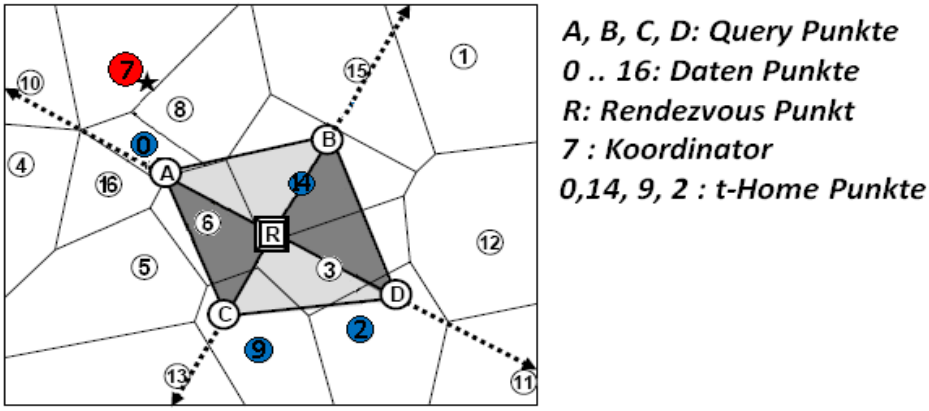

Fig.6. Partition des dimensionalen Raumes (angelehnt an [6])

2. DSS – Interne Skylines Suche

Der Algorithmus sucht nach internen Skylines in jedem Dreieck (ARC, ARB, BRD und CRD). Der t-Home Punkt eines Dreiecks wird automatisch einen Punkt der Spatial Skyline in diese Dreieck sein. Die ganze gefundene Spatial Skyline werden über den t-Home Punkt verwaltet.

Danach wird der t-Home Punkt unbekannte Skyline Punkte zwischen seinen Voronoi-Nachbarn suchen, dafür müssen sich die Voronoi-Nachbarn in dem Dreieck oder in seinem erweiterten Bereich befinden und der jeweilige Voronoi-Bereich mit der Convex Hull schneiden.

Zum Beispiel in Fig. 6 der t-Home Punkt 0 kann nur der Punkt 8 berücksichtigen. Der Punkt befindet sich nicht in der Dreieck ARD aber schon in der erweitere Bereich des Dreiecks, und der Voronoi Bereich der Punkt 8 schneidet sich mit der Convex Hull.

Somit, die internen Skyline Punkte befinden sich innerhalb der Convex Hull oder auch außerhalb, sofern sich der jeweilige Voronoi Bereich mit der Convex Hull schneidet.

3. DSS – Externe Skylines Suche

Die Suche externer Skylines wird in den erweiterten Bereichen der Dreiecke realisiert.

Jeder interne Skyline Punkt sucht unbekannte Skyline Punkte zwischen seinen Voronoi Nachbarn. Die Nachbarn müssen dem gleichen Dreieck des internen Punktes entsprechen und sich außerhalb der Convex Hull befinden.

Die Information jede neue Skyline Punt wird an der t-Home Punkt geschickt, damit er die ganze Spatial Skyline bildet.

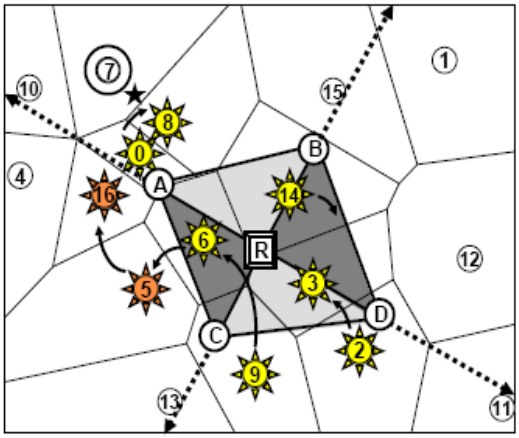

Fig. 7. Skylines Suche [6]

In der Fig. 6 ist der Punkt 9 t-Home Punkt des ARC Dreiecks, deswegen ein interner Punkt auch, in der internen Suche wurde der Punkt 6 gefunden und in der externen Suche wurden die Punkte 5 und 16 gefunden.

Sowohl die interne als auch die externe Suche in jedem Dreiecks Bereich können parallel realisiert werden.

Nach der Realisierung der Interne und externe Suche wird jede t-Home Punkt seine Spatial Skyline an den Rendezvous Punkt geschickt, damit er die gesamte und finale Skyline bildet.

3.2. Bewertung

Die Spatial Skyline Queries nutzen die räumlich Position der Datenobjekte und Query Punkte aus, um die Skyline Punkte zu finden. Besonders der Distributed Spatial Skyline (DSS) Algorithmus nach den drei Partitionierungen ermöglicht eine gründliche Suche. Der Besuch aller Punkte ist nicht notwendig, weil nach der Partitionierung die Punkte teilweise schon klassifiziert sind.

Der DSS Algorithmus wurde mit einem zentralisierten Algorithmus in einem Netz mit 292 Knoten verglichen. Die Ergebnisse zeigen, dass DSS 68% weniger Kommunikation-Overhead hat. Außerdem skaliert der DSS Algorithmus besser, weil die Kommunikation-Overhead in Bezug auf die Größe des Netzes leichter ansteigt als ein zentralisierter Algorithmus.

Bezüglich der Skyline Suche hat der DSS Algorithmus die gleiche Genauigkeit wie in einem zentralisierten Algorithmus. Und dank der parallelen Suche in den einzelnen Bereichen kann man partielle Ergebnisse progressiv erhalten. Daher wird das finale Ergebnis schneller erreicht. Es bedeutet, dass weniger Laufzeit benötigt wird.

Folglich kann man feststellen, dass der DSS Algorithmus die Leistung der zentralisierten Algorithmen für Spatial Skyline erhöht, ohne die Genauigkeit der Ergebnisse zu vermindern.

4. Zusammenfassung

In dieser Arbeit wurden zwei Ansätze präsentiert, beide zeigen unterschiedliche Anwendungen für Skyline Queries. Daher gibt es verschiedene Algorithmen, um die Skyline Space zu steuern, sowie die Skyline Punkte zu finden.

Beim ersten kann man beweisen, dass Skyline Queries geeignet sind für die Suche von Top-k Anfrageergebnissen, insbesondere mit n Dimensionen. Der Algorithmus benutz ein Compressed Skycube, damit die Suche und Aktualisierung der Daten effektiver wird. Die Eigenschaften des Compressed Skycubes zeigt, dass Skyline Queries geeignet sind für dynamische Umgebungen, wie P2P Netze.

Beim zweiten Ansatz werden Skyline Queries in andere Anwendungsumgebung benutzt wie Wireless Sensor Network. Man sucht eine einzige Spatial Skyline bezüglich einer gesamten Gruppe der Anfragen. Der entsprechende Algorithmus heißt Distributed Spatial Skyline (DSS) und ist ein verteilter Algorithmus für Skyline-Suche.

Mit der Auswahl der Ansätze wurde versucht zu beweisen, dass Skyline Queries geeignet sind für dynamische und verteilte Umgebungen. Folglich repräsentieren sie einen guten Mechanismus für Query Processing in P2P Datenbanken.

5. Ausblick

Da es noch keine explizite Methode für P2P Umgebungen gibt, ist das Ziel, eine Skyline Query Processing Methode für eine strukturierte P2P Umgebung zu implementieren.

Ein Framework für Skyline Query Processing soll implementiert werden. Die Algorithmen müssen nicht nur eine Anfrage beantworten, sondern auch einen effizienten und optimalen Vorgang realisieren. Dazu sollten folgende Faktoren berücksichtigt werden: die Antwortzeit, niedrige Kommunikationskosten und gleiche Arbeitslast in jedem Peer, um eine Anfrage zu verarbeiten.

Der erste Schritt ist der Aufbau einer P2P Testumgebung: ein simuliertes P2P Netz für Chord wurde ausgewählt. Nach der Implementierung des Frameworks soll die neue Methode analysiert werden. Dazu wird ein Probe-Dataset benötigt. Das Probe-Dataset muss auch mit existierenden Methoden getestet werden und diese mit der neuen Methode verglichen werden.

Nach den Tests ist das Ziel das Framework in einem echten P2P Netz anzuwenden und mit realen Daten zu testen.

A. Literatur

[1] J. Cusi Juarez. *P2P Datenbanken.* Ausarbeitung im Rahmen der Vorlesung Anwendungen I im Studiengang Informatik Master of Science am Studiendepartment Informatik der Fakultät. Technik und Informatik der Hochschule für Angewandte Wissenschaften Hamburg. Februar 2010.

[2] D. Papadias, Y. Tao, G. Fu, B. Seeger. *An Optimal and Progressive Algorithm for Skyline Queries*. In ACM SIGMOD 2003, June 9-12.

[3] Y. Yuan, X. Lin, Q. Liu, W. Wang, J. X. Yu, Q. Zhang. *Efficient Computation of the Skyline Cube*. In VLDB 2005, pages 241–252, 2005.

[4] T. Xia, D. Zhang. *Refreshing the sky: The compressing skycube with efficient support for frequent updates*. In SIGMOD, 2006.

[5] J. Lee, G. You, I. Sohn, S. Hwang, K. Ko, Z. Lee. *Supporting Personalized Top-k Skyline Queries using Partial Compressed Skycube*. In ACM WIDM‘07, 2007.

[6] S. Yoon, C. Shahabi. *Poster Abstract: A Distributed Algorithm to Compute Spatial Skyline in Wireless Sensor Networks.* In ACM IPSN‘09, April 2009.

[7] http://de.wikipedia.org/wiki/Voronoi-Diagramm

[8] Y. Luo. *Multi-Dimensional Skyline Query Processing*. Dissertation submitted in fulfillment of the requirements for the degree of Master of Computer Science School of Computer Science and Engineering the University of New South Wales Sydney, Australia. August 2004.

B. Abbildungsverzeichnis