Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung - Anwendungen 1

Julissa Cusi Juarez

P2P Datenbanken

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Julissa Cusi Juarez

P2P Datenbanken

Ausarbeitung eingereicht 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

Betreuer: Prof. Dr. Olaf Zukunft

Abgegeben am 27.02.2010

Inhalt

1. Einleitung
2. Grundlagen
   2.1. Klassische Datenbanksysteme
   2.2. Verteilte Datenbanksysteme
      2.2.1. Klassifikation
      2.2.2. Fragmentierung und Replikation
   2.3. Peer-to-Peer (P2P)
      2.3.1. Eigenschaften
      2.3.2. Unstrukturierte P2P Netze
      2.3.3. Strukturierte P2P Netze

3. P2P Datenbanksysteme
   3.1. Definition
   3.2. Abgrenzung zu P2P Dateisystemen
   3.3. Bewertung

4. Query Processing in P2P Datenbanksysteme
   4.1. Mögliche Datenmodelle und Anfragesprachen
   4.2. Abhängigkeiten vom verfügbaren Index
   4.3. Fallstudie: Skyframe
      4.3.1. Skyline Query
      4.3.2. Architektur

5. Zusammenfassung
6. Ausblick

A. Literatur
B. Abbildungsverzeichnis

1. Einleitung

P2P Systeme haben sich infolge vieler Gründe entwickelt.

Einerseits gibt es die Entwicklung der Endbenutzer-Kapazitäten, das heißt, viele Benutzern haben Zuhause hochwertige Computer mit hoher Rechenleistung und hoher Speicherkapazität. Andererseits gibt es die Entwicklung der Kommunikation-Netzwerke. Die Internetverbindung ist immer mehr verfügbar, mittels Breitbandverbindung, Flatrate, usw.

Diese Gründe repräsentieren eine verfügbare technische Infrastruktur. Darüber hinaus existiert einen hohen Bedarf einer hohen Anzahl von Benutzern weltweit, Information auszutauschen, durch Content Sharing Websysteme.

2. Grundlagen

2.1. Klassische Datenbanksysteme

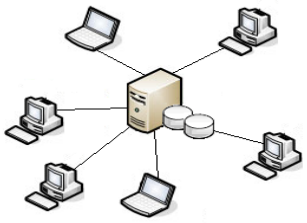


Abbildung 1. Zentralisierte DBS

Die klassischen Datenbanksysteme [11] zeichnen sich durch ein Client/Server basierendes Modell mit einem zentralisierten Server aus. Die passen gut zu zentralisierten Unternehmen. da sie eine zentrale Administration des Servers ermöglichen.

Normalerweise steuern klassische Datenbanksysteme relationale Datenbanken und benutzen SQL als Anfragesprache.

2.2. Verteilte Datenbanksysteme

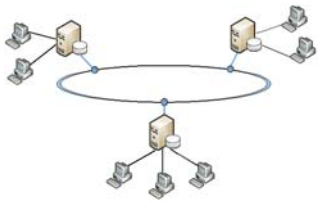
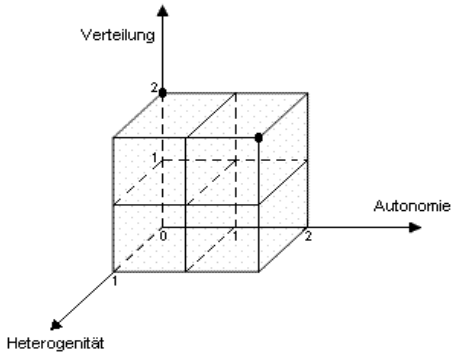


Abbildung 2. Verteilte DBS

Ein verteiltes Datenbankmanagementsystem ist eine Software, die eine oder mehrere logisch zusammenhängende Datenbanken über ein Netzwerk steuert [1].

2.2.1. Klassifikation

Es gibt eine Klassifikation nach drei Dimensionen, die haben unterschiedliche Werte. Die verschiedene Architekturalternativen von DBS können durch diese Dimensionen charakterisiert werden.



Dimension Autonomie	0: Vollständige Integration 1: Semi-autonome System 2: Vollständige Autonomie
Dimension Verteilung	0: Nicht verteilt (zentralisiert) 1: Client / Server System 2: Verteilung Peer-to-Peer
Dimension Heterogenität	0: Homogenes System 1: Heterogenes System

Abbildung 3. Klassifikation DBS [11]

Der Punkt (A_0, V_0, H_0) entspricht einem zentralen DBS. Der Punkt (A_0, V_2, H_0) entspricht einem verteilten DBS [11].

2.2.2. Fragmentierung und Replikation

Fragmentierung ist die Verteilung der Daten auf Knoten. Es gibt drei Möglichkeiten, relational organisierte Daten zu fragmentieren:

- Horizontal (nach Zeilen)
- Vertikal (nach Spalten)
- Hybrid

Replikation ist eine absichtliche redundante Speicherung der Daten auf den Knoten. Auch hier gibt es drei Möglichkeiten:

- Vollständige Replikation aller Daten
- Partielle Replikation einiger ausgewählter Daten
- Keine Replikation

2.3. Peer-to-Peer (P2P)

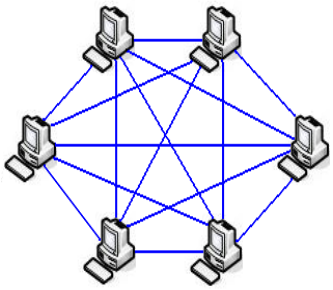


Abbildung 4. P2P System

Ein Peer-to-Peer System ist eine Overlay Architektur auf einem physischen Netzwerk. Ursprünglich wurde es für File-sharing-Anwendungen entwickelt.

Ein P2P- System ist dezentralisiert und die Knoten (Peers) sind gleichberechtigt und autonom.

In einem P2P System kann ein Peer jederzeit in das Netz eintreten oder es verlassen, deswegen ist ein P2P System ein selbst organisiertes und dynamisches System.

Kein Peer hat einen Überblick über das gesamte System, nur über die Peers mit denen er interagiert.

Ein P2P System benutzt verteilte und geteilte Ressourcen.

2.3.1. Eigenschaften

Das P2P Paradigma hat folgende Eigenschaften [1,5]:

- Skalierbarkeit bezüglich der Knotennummer, die Anzahl von Benutzern kann die Anzahl von Internet-Benutzern sein.

- Dezentralisiert in der reinsten Form von P2P Netzen, gibt es kein Konzept von „Server“. Jeder Knoten hat Client- sowie Servereigenschaften.

- Autonomie bezüglich mehrerer Aspekte.

    - Nach Speicherung, da ein Peer nur ihre gewünschte Information, speichert.
    - Nach Ausführung, weil der Peer Anfrage freiwillig beantwortet und seine eigene Daten verändert.

  - Bezüglich Standzeit, die Peers bestimmen, wie viel Zeit sie in dem System bleiben.
  - Und letztendlich nach Verbindung, jeder Peer entscheidet mit welchen Knoten wird er sich verbinden.

- Direkter Zugriff auf Daten in der eigenen Quelle, weil die Daten nicht zentralisiert sind.

- Robustheit und Härtung gegen Peer-Ausfälle

- Einfache Entwicklung, weil die Knoten keine besondere Infrastruktur brauchen.

2.3.2. Unstrukturierte P2P Netze

Es gibt drei Typen unstrukturierter P2P-Netze:

- **Zentralisierte P2P-Systeme**. Hier gibt es einen zentralen Peer, der den Inhalt von allen Peers kennt. Eine Anfrage wird an den zentralen Peer geschickt, dann schickt er die Anfrage an die geeigneten Peers weiter. Die Antwort wird direkt an den Peer, der die Anfrage gemacht hat, beantwortet. Ein Repräsentant dieses Ansatzes ist das Beispielsystem Napster.

- **Reine P2P-Systeme**. Hier gibt es keinen zentralen Peer. Eine Anfrage wird im Netz durch Flutung verbreitet und wer die gewünschte Information hat, beantwortet die Anfrage durch die Overlaylinks. Ein Repräsentant dieses Ansatzes ist das Beispielsystem Gnutella 0.4.

- **Hybride P2P-Systeme**. Dies ist eine Kombination von zentralisierten und reinen P2P Systeme. Man hat Superpeers und Peers, die Superpeers sind wie in ein reines P2P System miteinander verbunden und eine Gruppe von Peers bilden ein zentralisiertes P2P System mit einem Superpeer aus. Ein Repräsentant dieses Ansatzes ist das Beispielsystem Gnutella 0.6.

2.3.3. Strukturierte P2P Netze

Ein strukturiertes P2P System benutzt Distributed Hash Table (DHT), um die Peers zu organisieren und zu indexieren. Jeder Peer ist verantwortlich für einen bestimmten Teil des Netzwerkgehaltes.

Um den verantwortlichen Peer für einen bestimmten Inhalt zu bestimmen, wird ein Protokoll benutzt. Repräsentanten dieses Ansatzes sind CAN, Chord [5].

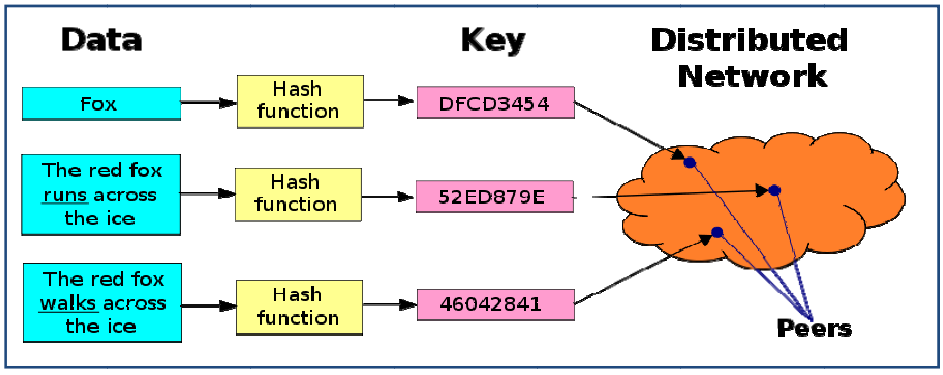


Abbildung 5. Distributed Hash Tables [7]

3. P2P Datenbanksysteme

3.1. Definition

Es ist eine Kollektion von den lokalen autonomen Datenbehältern (Peers), die miteinander interagieren [1].

Die P2P Datenbanksysteme kombinieren die dezentralisierten Eigenschaften und die Autonomie der P2P Systeme mit der Semantik der VDBS.

Eine besondere Eigenschaft ist die Vermeidung von globalen Datenbankschemata, anstatt dessen existiert ein Mapping Graph, der die Verbindungen zwischen den Peers repräsentiert. Dementsprechend hat jeder Peer ein lokales Schema, ein Export-Schema, wo die geteilten Daten repräsentiert sind, und der Mapping Graph. (s. Abbildung 6)

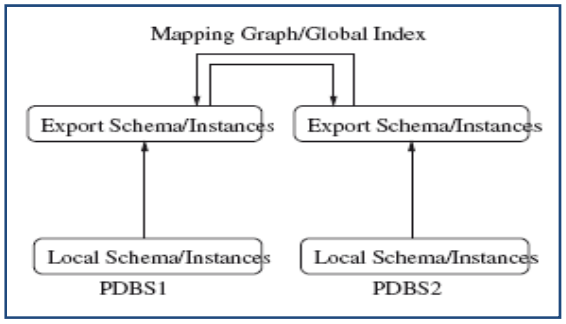


Abbildung 6. Integrationarchitektur der Daten [1]

3.2. Abgrenzung zu P2P Dateisystemen

Ein P2P System ermöglicht mehrere Anwendungen, eine davon ist die P2P Dateisystem. Die P2P Dateisystemen ermöglichen Dateien entfernt zu speichern und auf diese entfernt zuzugreifen, als ob sie lokal wären [9,12].

P2P Dateisysteme und P2P Datenbanksysteme erben Eigenschaften von P2P Systeme, wie Transparenz, Skalierbarkeit, Dezentralisation, Autonomie, usw. sowie Vorteile und Nachteile von P2P Systeme. Der wichtigste Unterschied zwischen P2P Dateisysteme und P2P Datenbanksysteme sind die Anfragekomplexität und Anfrageverarbeitung über die Information, die jeder Knoten teilt.

Die Indexierung in P2P Dateisysteme sind nicht über Relationen und Attributen wie in P2P Datenbanksysteme, sondern nur normalerweise ein einziges Attribut oder ein paar Stichworte der Datei. Das bedeutet, dass in ein P2P Dateisystem kann man keine komplexe Anfrage mit Attributen, Joins, usw. realisiert werden.

Beispiel Anfrage in P2P Dateisystem	Beispiel Anfrage in P2P Datenbanksystem
„Alle Bilderdateien von Hamburg“	„Alle Bilder von Hamburg in Winter, in Schwarzweiß mit Größe zwischen 1.0 MB und 1.5 MB, die in einer Ausstellung angezeigt wurden“

Um Anfragen zu realisieren, braucht jeder Peer nicht nur eine Indexierung sondern auch ein Schema zu haben, so kann man die verschiedenen Anfragen zwischen den Peers gemappt und übersetzt werden.

3.3. Bewertung

Vorteile von P2P Datenbanksysteme

- Die P2P-Vorteile wie Skalierbarkeit, Dezentralisation, Autonomie, bleiben erhalten.
- Eine wichtige Eigenschaft ist, dass keine besondere Infrastruktur in den Knoten benötigt wird.

Nachteile von P2P Datenbanksysteme

- Es kann passiert, dass einige Anfragen nicht beantwortet werden können, wegen des Austrittes eines Peers, der die gewünschten Daten hat.
- Eine Einschränkung für die Anfrageverarbeitung kommt vor, weil typischerweise SQL als Anfragesprache nicht unterstützt ist.

4. Query Processing in P2P Datenbanksysteme

Um eine Anfrage zu verarbeiten, muss die Anfrage durch das Overlaynetz verbreitet werden. Der Vorgang sowie die Anfragesprache hängen von der Topologie ab.

Man benutzt spezifischen Indexes, die Indexes werden überprüft, dann die Anfragen an die geeigneten Peers geschickt und dort bearbeitet.

In ein P2P System gibt es keine Garantie, alle Antworten zu erhalten, da ein geeigneter Peer das Netz verlassen kann oder viel Zeit um die Antwort zu geben brauchen kann. Daher benutzt man Replikation- und Laufzeittoleranztechniken.

4.1. Mögliche Datenmodelle und Anfragesprachen

Die Datenmodelle für P2P Datenbanken sowie die Anfragesprache hängen von der Art des P2P Systems ab.

Die Datenmodelle können sein:

- **Fixe Attributen**. Die Attribute können als Metadaten indexiert sein. Es ist nicht flexibel, da die ganze Struktur wegen einer Veränderung wieder gemacht werden muss.
- **Relationalmodell**. Man kann komplexe Anfrage machen aber man verliert P2P Eigenschaften wie Skalierbarkeit.
- **XML**. Arbeitet mit semi-strukturierten Daten. Dieses Modell erlaubt die Navigation in ein Dokument, um die Information zu erreichen.
- **RDF**. Hat ein XML-Format aber mit Struktur, die Struktur darf jeder definieren. Es ist ein Austauschformat, das in Semantik Web benutzt wird.

Bezüglich Anfragesprachen:

- **Keine**. Wenn man fixen Attributen hat. Die Anfrage wird mittels Lookup-queries realisiert und wird dann zu einem P2P Dateisystem.
- **Relational**. Es ist normalerweise eine Untermenge von SQL. Beispielsystem PeerDB [1].

- **XML**. Es gibt Anfragesprache wie XPath oder XQuery zu benutzen. Beispielsystem Galanis et al [1].

- **RDF**. Man kann SPARQL oder Logic Language als Anfragesprache benutzen. Beispielsystem GridVine [1].

4.2. Abhängigkeiten vom verfügbaren Index

Die Datenmodelle für P2P Datenbanken sowie die Anfragesprache hängen von der Art des P2P Systems ab. Es gibt drei Typen [5]:

- **Kein Index.** In diesen Fall die Anfrage wird durch Flutung in das Netz verbreitet. Dieser Typ entspricht reinen P2P Netze.

- **Zentralisierter Index.** In diesen Fall existiert ein Peer, mit Kenntnis der Inhalte aller Knoten, diese Eigenschaft erzeugt aber Bottleneck. Dieser Indextyp entspricht zentralisierten P2P Netze.

- **Verteilter Index.** Bezüglich der Overlaytopologie:

    - **In unstrukturierte Systeme**. Man benutzt sogenannten Routing Indexes, um aufwendige Flutung während der Anfrageverarbeitung zu vermeiden. Ein Peer hat Information über den Inhalt von erreichbaren Knoten. So kann man eine Repräsentation von erreichbaren Informationen über jede Nachbarn.

    - **In strukturierte Systeme**. Jeder Peer hat Indexinformation aller Daten in eine Verteilte Hash Tabelle (DHT).

4.3. Fallstudie: Skyframe

Skyframe ist ein Framework für effiziente Skyline Query Processing in P2P Systeme. Sein Ziel ist den Zeitverlauf von Anfrageverarbeitung zu optimieren, die Netzwerkkommunikationskosten zu reduzieren und die Query-load durch die Peers zu balancieren [6].

4.3.1. Skyline Query

Skyline Queries werden für Multi-Kriterien Entscheidungsunterstützung benutzt [6].

Ein Skyline Query gibt eine Menge von Datenobjekten aus der Datenbank zurück. Die ergebenen Datenobjekte heißen Skyline Punkte, die sich in einem n-dimensionalen Raum befinden. Jede Dimension entspricht einem Query-Attribut.

Ein Skyline Punkt wird für andere Datenobjekte nicht dominiert [6]. Ein Punkt dominiert andere Punkte wenn er die Skyline Query besser erfüllt, bezüglich aller Dimensionen.

Alle Skyline Punkte bilden die Skyline.

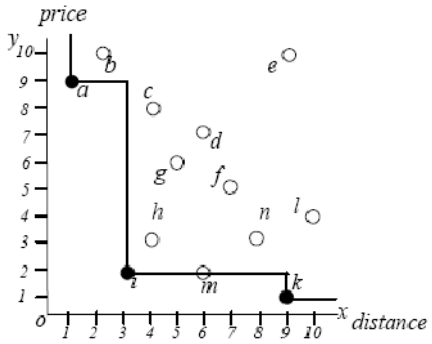


Abbildung 7. Beispiel Datenmenge und Skyline [10]

Die Abbildung 7 stellt Skyline Punkte für eine Skyline Query dar [10]. Man hat einer Menge von Hotels, die Entfernung zum Meer ist in *distance* Dimension repräsentiert, auf die gleiche Weise ist der Preis in *price* Dimension repräsentiert. Die Anfrage bietet die Hotels, die näher zum Meer und billiger sind. Die Punkte *a*, *l*, und *k* erfüllen die Anfrage besser als andere Punkte und bilden die Skyline für die Anfrage.

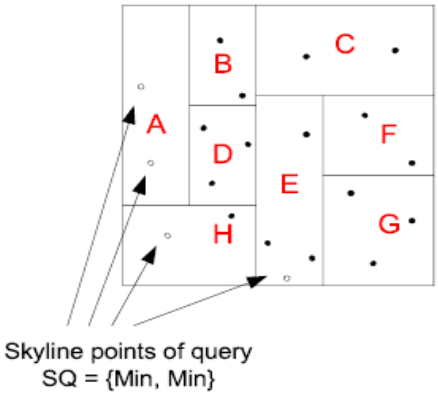


Abbildung 8. Beispiel von Skyline Query im zweidimensionalen Raum [6]

Die Abbildung 8 stellt Skyline Punkte, die in den Knoten A bis G verteilt sind, dar. Die Skyline Query *SQ* bietet die Minimumwerte in beide Dimensionen.

4.3.2. Architektur

1. **Skyframe.** Die folgende Abbildung fasst die Skyframe Architektur zusammen [6]. Die wichtigste Komponente sind Query Processing Manager, der die Skyline Query Processing übernimmt, und Load Balancing Manager, der für die Query Balancing unter den Knoten verantwortlich ist.

   Skyframe hat noch weitere supplementär Methoden, die von beiden Managern benutz werden.

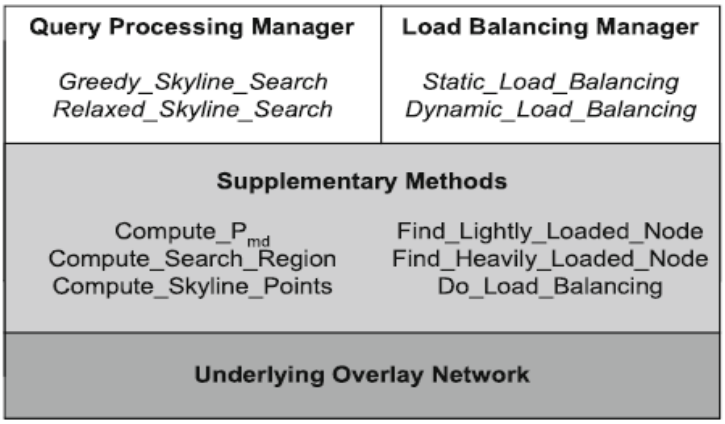


Abbildung 9. Skyframe [6]

2. **Query Processing Manager.** Das Ziel von Query Processing Manager ist der Suchraum zu beschneiden. Dafür hat er zwei Methoden.

   - **Greedy Skyline Search (GSS).** Die Suche beginnt mit der Bestimmung eines Anfangsknotens, der *Initiator* Knoten bzw. SQ-*Starter* heißt. Der SQ-Starter ist ein Knoten deren lokale Werte in der finalen Skyline unter Garantie sein werden.

     Für das Beispiel von Abbildung 8, der SQ-Starter wäre der Knoten H, weil die Skyline Query die Minimumwerte in beide Dimensionen bietet. Im weiteren Verlauf bestimmt man einen dominanten Punkt (p_{md}) im SQ-Starter Knoten (s. Abbildung 10).

     Im weiteren Verlauf man bestimmt der dominante Punkt in jedem Knoten, wenn dieser Punkt für p_{md} dominiert wird. Dann wird der entsprechende Knoten vom Suchraum entfernt.

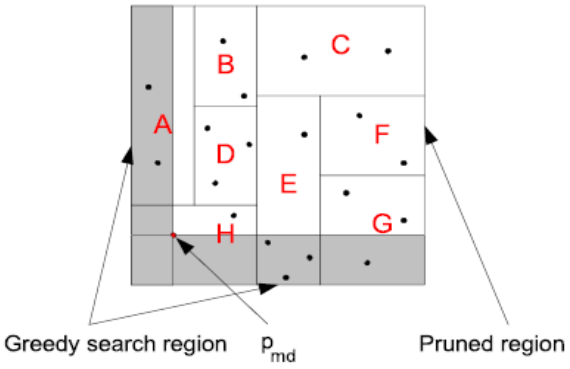


Abbildung 10. Greedy Skyline Search [6]

   - **Relaxed Skyline Search (RSS).** Die Suche hat zwei Phasen.

     Die erste Phase besteht in die Suche von *SQ-Borders*. Ein SQ-Border ist ein Knoten, der am Rand von Suchraum ist. Für das Beispiel von Abbildung 8, die SQ-Borders wäre die Knoten A, H, E und G, da die Skyline Query die Minimumwerte in beide Dimensionen bietet (s. Abbildung 11).

     Alle SQ-Border Knoten bestimmen seine lokale Skyline Punkte und ergeben die Punkte dem SQ-Starter Knoten.

Der Relaxed Skyline Search Raum besteht aus den SQ-Borders Regionen und dem Greedy Skyline Search Raum (s. Abbildung 11).

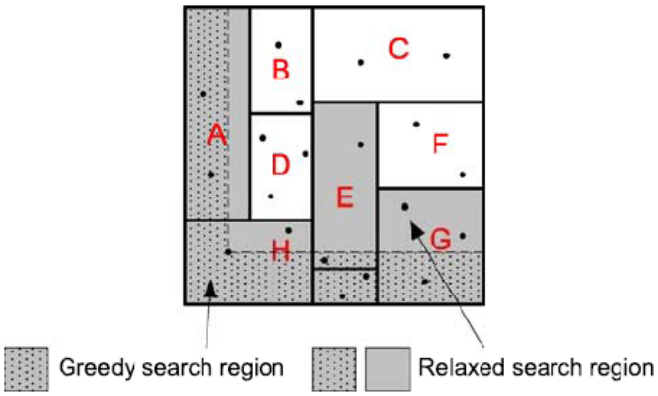


Abbildung 11. Relaxed Search Space [6]

Wenn der ganze Greedy Skyline Search Raum in den SQ-Borders Regionen nicht beinhalt wird, dann fängt die zweite Phase an.

Die zweite Phase sucht zusätzliche Regionen. Die Skyline Query wird zu Nachbarknoten den SQ-Border Knoten verbreitet. Diesen Knoten bestimmen auch seine lokale Skyline Punkte und ergeben die dem SQ-Starter Knoten. Wenn keine zusätzliche Region mehr nötig ist, dann bestimmt der SQ-Starter der globale Suchraum (s. Abbildung 12).

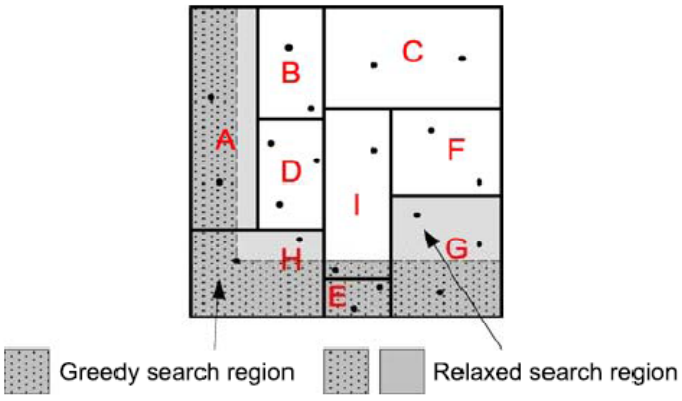


Abbildung 12. Relaxed Search Space mit zusätzlichen Schritten [6]

Die Greedy Skyline Search- und Relaxed Skyline Search Methoden funktionieren gleichseitig.

3. **Load-balancing Manager.** Der Load-balancing Manager ist verantwortlich für die Query Load Balancing zwischen den Knoten.

   Das Query Load eines Knotens ist die Summe von (1) die Menge der lokalen Datenpunkte, die ergeben sind, um die Anfrage zu beantworten, und (2) die Menge der Nachrichten, die beim Knoten verursacht wird, aber zu keiner lokalen Anfrageverarbeitung führt [6].

Der Load-balancing Manager hat zwei Methoden:

- **Query load balanced Partition.** Das System partitioniert den Daten Raum und das Query-Load wird auf die Partitionen gleichmäßig verteilt. Wonach geeignetester Knoten wird ausgewählt, um das Query-Load zu teilen in den Fall von Join- oder Departure-Knoten Prozess.

  In den Fall von Join-Knoten Prozess aussucht man einen Nachbarknoten, der das schwerste Query-Load hat, um die Joinpetition zu senden. Der neue Knoten kann den geeigneten Knoten zwischen mehrere mit denen er verbunden ist aussuchen.

  In den Fall von Departure-Knoten Prozess hat man zwei Möglichkeiten. Zuerst, wenn der Nachbarknoten ein leichtes Query-load hat, dann nimmt der Nachbarknoten das Load des ausgehenden Knotens. Andererseits die zweite Möglichkeit, wenn alle Nachbarknoten ein schweres Query-load haben, dann sucht der ausgehende Knoten andere Knoten *L*, der sowie als seine Nachbarknoten, leichtes Load hat. So kann der Knoten L der Position der ausgehende Knoten übernehmen.

- **Dynamic query load balancing**. In diese Methode überprüft zuerst jeder Knoten, ob es Loadbalance gibt. Wenn nicht muss eine Datenmigration, um das Query-load zu balancieren, anfangen.

  Wenn ein Knoten Ungleichgewicht bestimmt, hat er zwei Wege, um das Query-load zu balancieren. Erstens kann er mit einem Nachbarknoten ihre Datenräume wieder aufteilen. Falls der Knoten leicht geladen ist, muss er einen schwer geladenen Knoten suchen, und umgekehrt.

  Wenn ein Balancingprozess mit einem Nachbarknoten nicht genug ist, dann muss anderer Knoten ausgewählt werden, entweder einen leicht oder schwer geladen Knoten, nach dem Fall. Um den Balancingprozess zu realisieren der leicht geladene Knoten lässt seine Position und nimmt eine neue Position neben dem schwer geladenen Knoten.

5. Zusammenfassung

Die vorliegende Ausarbeitung präsentiert wichtigste Eigenschaften von P2P Systemen und vornehmlich P2P Datenbanken.

Ein P2P DBS kombiniert die dezentralisierten Eigenschaften und die Autonomie der P2P Systeme mit der Semantik des VDBS.

Das P2P Netz bestimmt die entsprechenden Datenmodelle und Anfragesprache von Datenbanken. Bezüglich Query Processing existieren verschiedene Methoden je nach Overlayarchitektur.

Eine wichtige Anwendung den P2P Systemen ist das P2P Dateisystem. Trotz erben P2P Dateisysteme und P2P Datenbanken gleiche Merkmale von P2P Systemen, gibt es Unterschiede und der wichtigste besteht aus Anfragekomplexität und Anfrageverarbeitung über die Information, die jeder Knoten teilt.

Schließlich wurde Skyframe, ein beispielhaftes Framework, beschreibt. Skyframe sucht den Zeitverlauf zu optimieren und niedrige Netzwerkkommunikationskosten zu erreichen.

6. Ausblick

Die Entwicklung eines Query-Processing Verfahrens für ein strukturiertes P2P System im Verlauf des Mastersstudiums ist geplant worden.

Um das Verfahren umzusetzen, muss eine Anwendung in typischen Szenarien entwickeln werden und letztendlich muss der entwickelten Verfahren mit einem existierenden Verfahren für das betrachtete Szenario verglichen werden.

A. Literatur

[1] A. Bonifati, P. K. Chrysanthis, A. M.Ouksel, K. Sattler. *Distributed Databases and Peer-to-Peer Databases: Past and Present*. In ACM SIGMOD Record, Vol.37 No.1, 2008.

[2] A. Eyal, A.Gal. *Self Organizing Semantic Topologies in P2P Data Integration Systems*. In IEEE International Conference on Data Engineering, 2009.

[3] X. Shen, Z. Li. *Implementing Database Management System in P2P Networks*. In International Seminar on Future Information Technology and Management Engineering, 2008.

[4] M. Karnstedt, K. Sattler, M. Haß, M. Hauswirth, B. Sapkota, R. Schmidt. *Estimating the Number of Answers with Guarantees for Structured Queries in P2P Databases*. In ACM Conference on Information and Knowledge Management CIKM´08, 2008.

[5] G. Koloniari, E. Pitoura. *Peer-to-Peer Management of XML Data: Issues and Research Challenges*. In ACM SIGMOD Record, Vol.34 No.2, 2005.

[6] S. Wang, Q. H. Vu, B. C. Ooi, A. K. H. Tung, L. Xu. *Skyframe: a framework for skyline query processing in peer-to-peer systems*. In VLDB Journal, Vol.18, No.1, 2009.

[7] http://en.wikipedia.org/wiki/Distributed_hash_table

[8] http://www.umkc.edu/is/security/p2p_explanation.asp

[9] M. Kabalkin. *Distributed File System.* Ausarbeitung 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, 2007

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

[11] M. Tamer Özsu, P. Valduriez. Principles *of Distributed Database Systems – Second Edition*. Prentice Hall, 1999, New Jersey.

[12] S. Ghemawat, H. Gobioff, S. Leung. *The Google File System*. In ACM SOSP´03, 2003.

B. Abbildungsverzeichnis

Abbildung 1. Zentralisierte DBS
Abbildung 2. Verteilte DBS
Abbildung 3. Klassifikation DBS
Abbildung 4. P2P System
Abbildung 5. Distributed Hash Tables
Abbildung 6. Integrationarchitektur der Daten
Abbildung 7. Beispiel Datenmenge und Skyline
Abbildung 8. Beispiel von Skyline Query im zweidimensionalen Raum
Abbildung 9. Skyframe
Abbildung 10. Greedy Skyline Search
Abbildung 11. Relaxed Search Space
Abbildung 12. Relaxed Search Space mit zusätzlichen Schritten