TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Übung zur Vorlesung *Einsatz und Realisierung von Datenbanksystemen* im SoSe18

Alexander van Renen, Maximilian E. Schüle (i3erdb@in.tum.de)
http://db.in.tum.de/teaching/ss18/impldb/

Blatt Nr. 05

Hausaufgabe 1

Gegeben sei eine verteilte Datenbank mit 8 Knoten, welche das *Quorum-Consensus*-Verfahren verwendet. 6 Knoten haben jeweils ein Gewicht von 1, 1 Knoten hat ein Gewicht von 2, 1 Knoten ein Gewicht von 3. Geben Sie alle gültigen Schreibquoren und zugehörige Lesequoren an.

Lösung:
Die Summe der Gewichte ist 11, ein Schreibquorum muss mehr als die Hälfte aller Gewichte vereinen.

- $Q_w = 6$ und $Q_r = 6$
- $Q_w = 7$ und $Q_r = 5$
- $Q_w = 8$ und $Q_r = 4$
- $Q_w = 9$ und $Q_r = 3$
- $Q_w = 10$ und $Q_r = 2$
- $Q_w = 11$ und $Q_r = 1$

Hausaufgabe 2

Zeigen Sie, dass die *write-all / read-any* Methode zur Synchronisation replizierter Daten einen Spezialfall der *Quorum-Consensus*-Methode darstellt.

- Für welche Art von Workloads eignet sich dieses Verfahren besonders gut?
- Wie werden Stimmen zugeordnet um *write-all / read-any* zu simulieren?
- Wie müssen die Quoren Q_w und Q_r vergeben werden?

Dieses Verfahren fordert einen sehr großen Aufwand beim Schreiben, aber nur minimalen Aufwand beim Lesen. Daher eignet es sich besonders gut für Workloads in denen wesentlich mehr Daten gelesen als geschrieben werden. Dies entspricht z.B. analytischen Anfragen.
Siehe Übungsbuch.

Hausaufgabe 3 Um Ausfallsicherheit zu garantieren ist ein Datenwert 'A' auf vier Rechnern verteilt. Jeder Rechner hält dabei eine vollständige Kopie von 'A'. Um Konsistenz zu garantieren wird das Quorum-Consensus-Verfahren eingesetzt. Dabei ist jedem Rechner ein Gewicht $w_i(A)$ wie folgt zugewiesen:

Rechner	Kopie	Gewicht
R_1	A_1	3
R_2	A_2	1
R_3	A_3	2
R_4	A_4	2

Das Lesequorum ist $Q_r(A) = 4$ und das Schreibquorum is $Q_w(A) = 5$.

a) Geben Sie **alle** Lesemöglichkeiten für eine Transaktion auf dem Datum 'A' nach dem Quorum-Consensus-Protokoll an.
A_1,A_2
A_1,A_2,A_3
A_1,A_2,A_3,A_4
A_1,A_2,A_4
A_1,A_3
A_1,A_3,A_4
A_1,A_4
A_2,A_3,A_4
A_3,A_4

b) Geben Sie **alle** Schreibmöglichkeiten für eine Transaktion auf dem Datum 'A' nach dem Quorum-Consensus-Protokoll an.
A_1,A_2,A_3
A_1,A_2,A_3,A_4
A_1,A_2,A_4
A_1,A_3
A_1,A_3,A_4
A_1,A_4
A_2,A_3,A_4

c) Zeigen Sie für dieses Beispiel, dass während eine Transaktion T_1 ein Schreibquorum auf A hält es für andere Transaktionen T_x nicht möglich ist ein Lesequorum für A zu bekommen.
Zum Schreiben muss die Transaktion T_1 mindestens Kopien mit einem Gesamtgewicht von 5 gesperrt haben. Insgesamt haben alle Kopien zusammen das Gewicht 8. Somit können maximal Kopien mit einem Gewicht von zusammen 3 übrig bleiben, womit das Lesequorum von 4 nicht mehr erfüllt werden kann.

Hausaufgabe 4

Überlegen Sie sich, welche Tupel bei der Anwendung des bloomfilterbasierten Joins in Abbildung 1 übertragen werden. Markieren Sie insbesondere, welche Tupel übertragen werden, obwohl sie keinen Joinpartner finden (sog. *false drops*). Wie kann die Anzahl dieser *false drops* verringert werden? Welche Eigenschaften sollte die Hashfunktion $h(c)$ die bei dieser Joinbearbeitung verwendet wird erfüllen?

- False Drops sind c7 und c8 die übertragen werden, obwohl sie keinen Joinpartner finden.
- Bei sinnvoller Hashfunktion weniger False Drops je größer der Vektor ist. Gute Hashfunktion! Mehrere Hashfunktionen zu benutzen verbessert auch die Effizienz des Bloomfilters
- Hashfunktion sollte möglichst gleichmäßig verteilen.

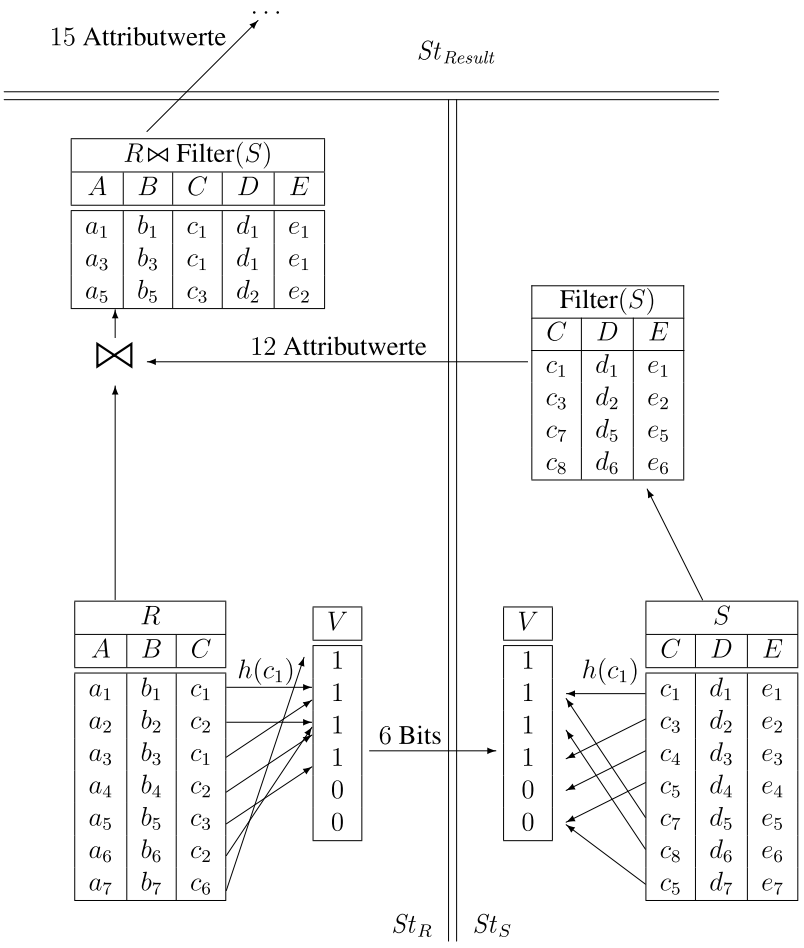

Abbildung 1: Beispiel einer verteilten Joinbearbeitung mit Bloomfilter.

Hausaufgabe 5

Zeigen Sie, dass die Suche in einem Chord-Overlaynetzwerk durch die Nutzung der FingerTabellen in maximal logarithmisch vielen Schritten zur Größe des Zahlenrings (bzw. der Anzahl der Stationen) durchgeführt werden kann. Verwenden Sie die Suche nach K57 beginnend an Station P11 (siehe Abbildung 2) zur Illustration.

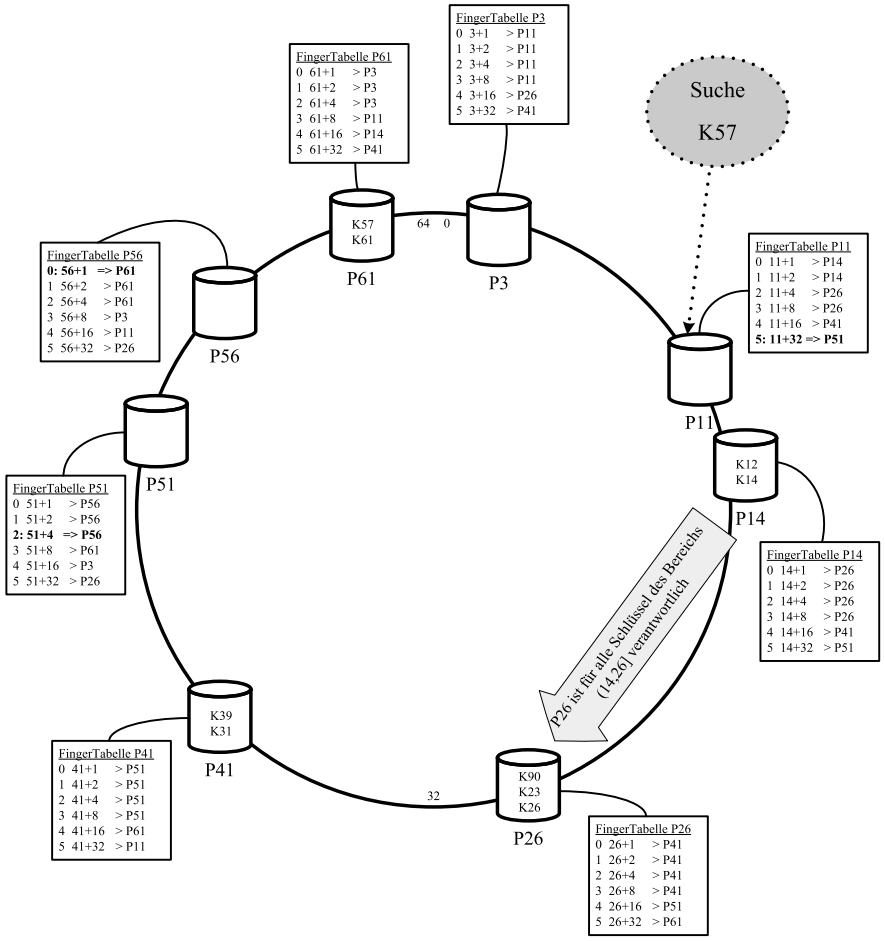

Abbildung 2: Beispiel eines Chord-Overlaynetzwerks.

Vgl. Übungsbuch.
z.z.: Mit jedem Sprung halbiert sich die Anzahl der Stationen.
Beweis durch vollständige Induktion über die Anzahl der Sprünge i.
Induktionsanfang: Sind wir direkt vor dem gesuchten Peer, also $i = 0$, müssen wir nicht mehr springen, es verbleiben $ln(1) = ln(2^0) = 0$ Sprünge.
Induktionsschritt: Vom i-ten zum $i+1$-Sprung halbiert sich die Anzahl verbleibender Stationen von 2^{n-i} nach 2^{n-i-1}
Induktionsvoraussetzung: Nach i Sprüngen reduziert sich die Anzahl verbleibender Stationen auf 2^{n-i}.
Wir wissen, dass sich innerhalb eines Intervalls $[p + 2^i, p + 2^{i+1})$ genau 2^i Stationen befinden, da wir sonst bereits zum nächsten Peer gesprungen wären. Damit haben wir bereits den Abstand auf 2^{n-i} Stationen halbiert (Induktionsvoraussetzung). Jetzt springen wir in das Intervall $[(p+2^i)+2^{i-1}, (p+2^i)+2^i)$ mit 2^{i-1} möglichen Stationen. Für den maximalen Abstand zum Ziel-Peer verbleiben nach dem $i + 1$-ten Sprung maximal $2^{n-(i+1)}$ Stationen, wenn wir $n - (i + 1)$ mal springen. Damit halbiert sich die Zahl verbleibender Sprünge auf $2^{n-(i+1)} = 2^{n-i-1}$

Hausaufgabe 6

Skizzieren Sie die Vorgehensweise beim Hinzufügen eines neuen Peers im Chord Netzwerk. Als Beispiel nehmen Sie die Hinzunahme eines Peers P33 in dem Beispiel-Netzwerk aus Abbildung 2.

Vgl Übungsbuch: Zuerst kontaktiert der neue Peer einen bekannten Peer im Chord Netzwerk. Dieser leitet ihn mittels der Hash-Funktion an den direkten Vorgänger des neuen Peers weiter. Von diesem übernimmt er die Fingertabelle und sein direkter Nachfolger wird ihm bekannt. Von diesem wiederum übernimmt er alle relevanten Daten, für die er nun zuständig ist.
Nach dem Einfügen eines neuen Peers Pk in ein Chord-Netzwerk müssen die Fingertabellen anderer Peers, „die bislang Pj referenziert haben, jetzt möglicherweise auf Pk geändert werden“ [a]. Dies passiert allerdings nicht beim Einfügen, sondern beim Stabilisieren [b]. Der Stabilisierungsalgorithmus wird in 30s Abständen aufgerufen [c], bzw. beim Einfügen eines neuen Peers [d]. Beim Einfügen wird nur der direkte Vorgänger über seinen neuen Nachfolger informiert.

[a] Kemper, Eickler: Datenbanksysteme, 10. Auflage
[b] `https://en.wikipedia.org/wiki/Chord_(peer-to-peer)#Pseudocode`
[c] `http://graal.ens-lyon.fr/~abenoit/reso06/papier/chord1.pdf`
[d] `http://sarwiki.informatik.hu-berlin.de/Chord#Eintritt_neuer_Knoten`