Informationsintegration

Kostenbasierte Optimierung verteilter Anfragen

- Summary -

Ulf Leser

Wissensmanagement in der Bioinformatik

Was bisher geschah ...

- Lokale Optimierung in einem DBMS
  - Optimierung von Anfragebäumen; Pushen von Join/Projektion; Benutzen von subsumierenden Anfragen
  - Überschaubar, da kaum *unkontrollierbare* Parameter

Und heute … ?

- Verteilte Optimierung für Anfragen über mehrere Quellen (DBMS)
  - Problem: Datenquellen
    - können ausfallen,
    - sich überlappen (mehrere Lösungswege!) und
    - haben unterschiedliche Performanz (Berechnung, als auch Latenz-/Übertragungszeiten)

Anfragebearbeitung - Unterschiede

- In zentralisierten DBMS
  - Ziel: Schnelle Antworten
  - Wesentlicher Faktor: IO
  - Heuristik: Zwischenergebnisse verringern
- In verteilten DBMS
  - Ziel: Schnelle Antworten
  - Wesentlicher Faktor: Netzwerk
  - Heuristik: Zwischenergebnisse verringern
  - Ressourcenverbrauch einzelner Knoten wird dabei u.U. aber erhöht

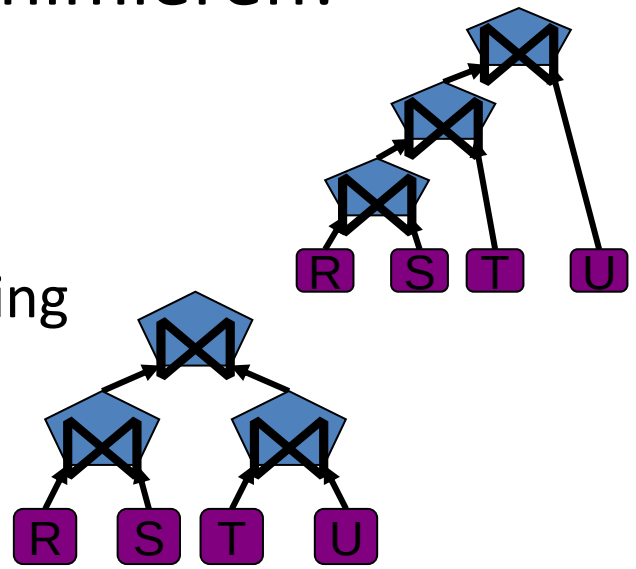
Allgemeine Ansätze zur Optimierung

- Multi Threading/Parallele Datentransfers
- Multicast vs Unicast
- Row blocking (blockweises Sammeln von Tupeln)
- (Result)-Caching
- IDs oder Tupel übertragen?

- Kostenabschätzung über Sampling (künstliche Anfragen) bzw Inline (Historie ausgeführter Anfragen); Regression

Join Order

- Jede globale Anfrage enthält Joins
- Die Reihenfolge der Ausführung der Joins bestimmt wesentlich die Menge an zu übertragenden Daten=>Minimieren!
- Problem: sehr(!) viele Pläne
- Lösung:
  - Heuristiken/Dynamic Programming



Dynamic Programming

- Idee
  - Gegeben sei ein Graph mit gewichteten Kanten
  - Wir wollen den kürzesten Pfad von A nach B finden
    - 1) Berechne alle direkten Kanten zu B
    - 2) Berechne alle Pfade der Länge 2 zu B
    - ...
    - n) Berechne alle Pfade der Länge n zu B
- Für Join order:
  - 1) Berechne besten Join für Paare von Relationen
  - 2) Berechne besten Join für Tripel von Relationen
  - ...
  - n-1) Berechne besten Join für alle n Relationen

Zusammenfassung

- Traffic minimieren
  - Grundproblem: Netzwerk anstatt Festplatte!
  - Join zu Quellen pushen (Es gibt natürlich Gegenbeispiele; abhängig von der Selektivität!)
  - Lokalität der Daten für Joinberechnung ausnutzen: Ort der Joinberechnung ist wichtig!