Fachpraktikum “Erweiterbare Datenbanksysteme” im WS 2015/16

Aufgabe 4 der Phase 2

Anfrageoptimierung für verteilte Datenbanken

auf Basis der Distributed Algebra

Ralf Hartmut Güting, Thomas Behr, Fabio Valdés, Holger Helmut Hennings

19.11.2015

Lehrgebiet Datenbanksysteme für neue Anwendungen
Fakultät für Mathematik und Informatik, Fernuniversität in Hagen

Aufgabe

- Verteilte Datenbanken und Anfrageauswertung möglich mit Secondo mit Hilfe der Distributed Algebra (genauer: Distributed2Algebra)
- Secondo-System (Master) kontrolliert viele Secondo-Server (Worker) auf verschiedenen Rechnern
- Master
    - verteilt Daten
    - veranlaßt parallele Berechnungen durch Worker (Abbildungen verteilter Daten)
    - sammelt verteilte Daten (Ergebnisse) wieder ein
- alles in ausführbarer Sprache
- Aufgabe: dies in SQL verfügbar machen (viel einfacher für Benutzer)

Distributed Algebra

- Zwei Ebenen
- Untere Ebene: Primitive
  - Verbinden mit Server (Worker)
  - beliebige Secondo-Befehle an einzelne Server schicken (Server über Nr. ansprechbar)
  - Secondo-Befehle zur parallelen Ausführung verschicken
  - Dateitransfer zwischen Servern bzw. Master/Server
  - Tupelströme partitionieren in Dateien
  - ...
- Obere Ebene: Abstraktion *verteilter Array* mit Operationen
  - Typen *darray*, *dfarray*, *dfmatrix*
  - *darray*: Felder enthalten Werte beliebiger Secondo-Typen (Relationen, Indexe, atomare Werte, ...). In Datenbanken der Worker gespeichert.
  - *dfarray*, *dfmatrix*: Felder enthalten Relationen. In Dateien der Worker gespeichert.

Distributed Algebra

Verteilter Array (*darray*, *dfarray*)

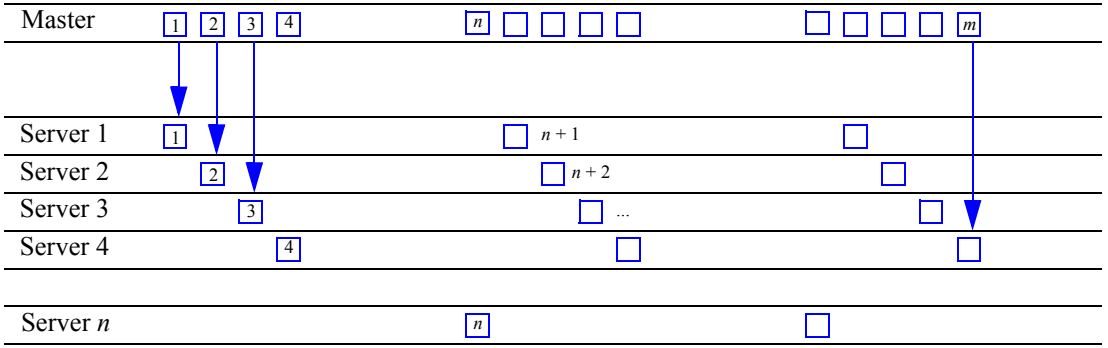

Figure 1: Creating a distributed array by partitioning data on the master.

Verteilte Matrix (*dfmatrix*)

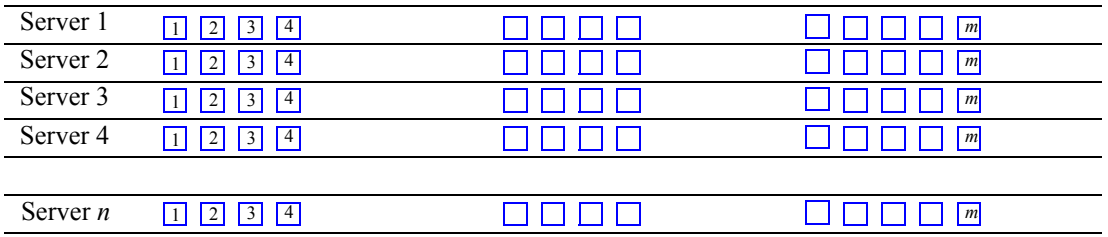

Figure 2: A distributed file matrix.

Distributed Algebra

Matrix kann spaltenweise in verteilten Array eingesammelt werden.

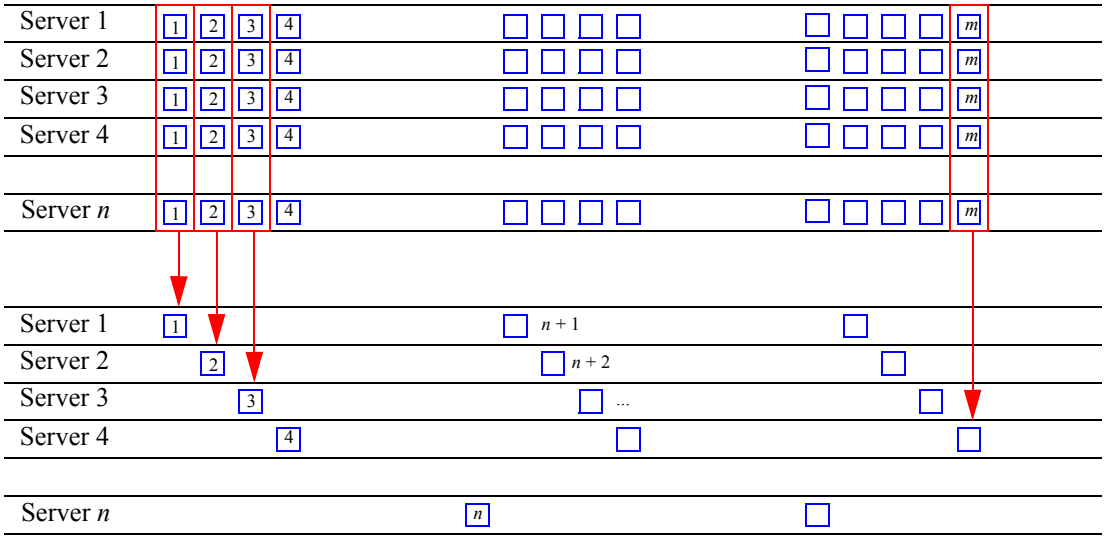

Figure 3: A distributed file matrix is collected into a distributed file array.

Distributed Algebra

- Obere Ebene: Operationen
  - **ddistribute**, **dfdistribute**: Tupelstrom verteilen; Ergebnis verteilter Array
  - **dloop**, **dmap**: jedes Feld bearbeiten (abbilden); Ergebnis neuer verteilter Array
  - **dloop2**, **dmap2**: für zwei Argumentarrays Felderpaare mit gleichem Index in Query bearbeiten (wichtig für Joins); Ergebnis neuer verteilter Array
  - **partition**: verteilten Array auf jedem Worker neu partitionieren (*dfarray* -> *dfmatrix*)
  - **collect2**: Matrix spaltenweise einsammeln (*dfmatrix* -> *dfarray*)
  - **areduce** , **areduce2**: eine oder zwei Matrizen dynamisch spaltenweise verarbeiten (*dfmatrix* -> *dfarray*, *dfmatrix* x *dfmatrix* -> *dfarray*)
  - **getValue**, **tie**: atomare Ergebnisse aus verteiltem Array auf den Master übertragen, dort aggregieren
  - **dsummarize**: Array von Relationen als Tupelstrom auf den Master bringen

Verteilte Datenbank

- Relationen können partitioniert oder repliziert verteilt sein
- in Datenbanken oder in Dateien
- atomare Objekte können repliziert sein
- Arten der Partitionierung
    - zufällig
        - sequentiell feste Größe
        - sequentiell round robin
    - nach Standardattribut (mittels Hashing)
    - nach Geoattribut (mit Gitterverteilung)

Beispieldatenbank: NRW als Shapefiles aus Openstreetmap

- ca. 6 Mio Gebäude
- ca. 1,5 Mio Straßen
- weiterhin Wasserwege, Landnutzung, Eisenbahnen, markante Punkte, Natur (Wälder usw..)

Arten von Anfragen

- nur Anfragen mit mindestens einer verteilten Relation
- Selektionen und Joins
- Ergebnisse zählen
- Ergebnisse einsammeln
- optional: Gruppieren und Aggregatfunktionen

Anfragen genauer

- Selektion Standardattribut
  - ohne Index
  - mit Index
- Selektion Geoattribut
  - ohne Index
  - mit Index
- Standard Equijoin
- Spatial Join
- Indexbasierter Equijoin
- Indexbasierter Spatial Join
- allgemeiner Join. Nur möglich, falls ein Argument verteilt, das andere repliziert vorliegt.

Zu tun

- Infrastruktur (Kenntnis des Optimierers über verteilte Datenbankobjekte und Worker)
  - Version 1: Prädikate werden manuell im Optimierer definiert
  - Version 2: Dynamische Prädikate werden mittels Anfragen an Secondo-Datenbank erzeugt. Dort werden diese Informationen in neuen Systemtabellen abgelegt.
- Überprüfen der Anfrage
  - sind Relationen bekannt?
  - wie verteilt?
  - sind atomare Objekte bekannt und repliziert?
- Kardinalitäten und Selektivitäten
  - Kardinalitäten verteilter Relationen stehen in Systemtabellen
  - Selektivitäten: Master besitzt verteilte Relation zumindest in Samplegröße. Ebenso erwähnte atomare Objekte. Deshalb können Selektivitäten wie bisher auf Master berechnet werden.
- Übersetzungsregeln
- Kostenfunktionen

Zu tun

- Nachbehandlung von Plänen
  - Selektionen sind in **dmap** übersetzt worden
  - **dmap** wenn möglich mit vorhergehendem oder nachfolgendem Operator verschmelzen