4. Übungsblatt

Ausgabe: 12. Juni 2015 · Besprechung: 18. Juni 2015

1 Parallele Hash Joins

Bei der Diskussion von Hash Joins waren wir bislang überwiegend von einer sequentiellen Implementierung ausgegangen.

Wie kann man Hash Joins (*radix joins*) effizient parallelisieren?

Überlegen Sie sich zunächst eine geeignete Strategie zur Parallelisierung von Hash Joins. Integrieren Sie diese Strategie dann in Ihre Joinimplementation (von Aufgabenblatt 3).

1.1 Paralleles Partitionieren

Besonders kritisch ist bei der Parallelisierung die Partitionierungs-Phase des Algorithmus. In der Vorlesung wurde am Rande skizziert, wie sich der einfach Hash Join parallelisieren lässt. Gelten die gleichen Voraussetzungen auch für *radix join*? Welches Verhalten Ihres Algorithmus erwarten Sie, wenn die Daten nicht gleichverteilt sind?

1.2 Parallele Joinverarbeitung

Nach der Partitionierung ergibt sich ganz natürlich die Möglichkeit zur parallelen Verarbeitung, weil die einzelnen Teil-Joins $r_i \bowtie s_i$ voneinander unabhängig berechnet werden können.

Wie könnten diese Teil-Joins möglichst geschickt auf vorhandene CPU-Ressourcen verteilt werden? Welches Verhalten erwarten Sie bei nicht-gleichverteilten Daten?

1.3 NUMA-Architekturen

In der Vorlesung wurden zwischenzeitlich NUMA-Architekturen diskutiert. Welches Verhalten erwarten Sie bei Ihrer Implementierung in Bezug auf NUMA-Effizienz? Überlegen Sie sich ob/wie sich Hash Joins besonders effizient in NUMA-Systemen verarbeiten lassen könnten.