MapReduce – Parallelität im Großen und im Kleinen

Das MapReduce-Programmiermodell wurde entworfen, um den Entwurf und die Implementierung paralleler Programme zu vereinfachen. Es basiert darauf, das Problem in eine Map-Phase und in eine Reduce-Phase aufzuteilen. Die vom Benutzer zu definierenden Funktionen *map* und *reduce* werden einzeln auf die Eingabedatenelemente angewandt. Da so das Problem in viele voneinander unabhängige Teilprobleme unterteilt wird, kann das Programm durch ein MapReduce-Framework parallel ausgeführt werden. Das MapReduce-Framework sorgt außerdem für die passende Verteilung der Daten auf die Rechenknoten, für Lastbalancierung, für die Kommunikation zwischen den Rechenknoten, ggf. Fehlertoleranz und viele weitere Dinge, mit denen sich ein Programmierer nicht gern aufhält.

Das Modell selbst entstammt der funktionalen Programmierung. Die Funktion *map* wendet eine benutzerdefinierte Funktion auf jedem Element eines Vektors (Arrays) an. Die Funktion *reduce* erzeugt – ebenfalls durch eine benutzerdefinierte Funktion angewandt auf alle Vektorelemente – ein Skalar aus diesen. Ihre Hintereinanderausführung bildet die Grundlage des MapReduce-Programmiermodells. Durch die flexible Implementierung der Map- und der Reduce-Funktion lässt sich eine erstaunliche Vielzahl von Problemen effizient mit nur diesen beiden Schritten lösen.

Ein Algorithmus in der MapReduce-Notation ausgedrückt skaliert durch seine hohe Parallelität sehr gut: Er läuft sowohl auf dem heimischen Dual-Core-PC als auch auf einem Cluster mit tausenden Knoten effizient. Für alle wichtigen Plattformen existieren entsprechende Frameworks:

- Ursprünglich aus dem Umfeld großer Server-Farmen stammend, in denen Terabyte an Daten verarbeitet werden, besitzt MapReduce dort auch heute noch große Bedeutung. Für jedermann bieten inzwischen verschiedene Cloud-Dienste einen MapReduce-Service mit dem auf Java basierenden Hadoop-Framework an.
- Das Phoenix-Framework, das auf PThreads basiert, zielt auf den Einsatz auf Multicore-Clustern.
- Für das Wissenschaftliche Rechnen auf Clustern mit verteiltem Speicher ist das Framework MR-MPI, das seine Kommunikation über MPI abwickelt, geeignet.
- Für die Parallelisierung kleinerer Anwendung auf Arbeitsplatzrechnern stellt z. B. das Qt-Framework eine an das MapReduce-Modell angelehnte C++-Klasse zur Verfügung.

Der Vortrag stellt zunächst das Programmiermodell und die theoretischen Grundlagen vor. An verschiedenen Problemen wird gezeigt, wie sich Algorithmen im MapReduce-Modell ausdrücken lassen. Weiter wird im Vortrag eine Übersicht über die MapReduce-Implementierungen für die verschiedenen Plattformen gegeben. Dabei wird auch auf Besonderheiten der Implementierungen sowie verschiedene Optimierungen eingegangen. Zuletzt wird der Einsatz anhand eines kleinen Beispielprogramms praktisch demonstriert.

Für das Verständnis des Vortrags sind Programmierkenntnisse notwendig.