Praktikum: Experimentelle Algorithmik — Sommersemester 2016

Berechnung von List Rank mit dem Big Data Framework Thrill

Lucas Berghäuser, Tom Zügel

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Einleitung

Spätestens durch den Erfolg von Hadoop ist sichtbar geworden, dass ein hoher Bedarf an Big Data Frameworks besteht. Eine der Erfolgsfaktoren ist der Highlevel-Ansatz dieser Frameworks, der die Implementierung von parallelen Algorithmen erleichtert, indem sämtliche Details durch das Framework übernommen werden. Das bedeutet, dass der Anwender weder die zugrundeliegende Topographie kennen und sich nicht um Themen wie Fehlerbehandlung kümmern muss. Dadurch werden diese Frameworks gerade für Big Data Analysen interessant.
Das Framework Thrill, welches sich noch in einem experimentellen Zustand befindet, versucht die Vorteile eines Highlevel-Ansatzes mit der Effizienz der Programmiersprache C++ zu verbinden. Hierbei wurden sämtliche Konzepte auf den neuesten Standard (C++14) ausgelegt. Intern verwendet Thrill das Message Passing Interface (MPI) für die Kommunikation zwischen den teilnehmenden Knoten des Clusters, welches einen Standard für verteiltes Rechnen darstellt.
Im Rahmen des Praktikums wurde List Rank mit Hilfe des Thrill-Frameworks implementiert und auf Effizienz untersucht.

Thrill

Eines der grundlegenden Konzepte in Thrill ist die Nutzung des Datentyps DIA (distributed immutable array), welche eine umfangreiche Schnittstelle zur Datenverarbeitung bietet. Die meisten Methoden, die das Framework dem Nutzer bietet, arbeiten auf diesem Datentyp (bzw. geben ein solches zurück). Zu diesen Methoden gehören zum Beispiel Map und Reduce, welche auch in anderen Big Data Frameworks (Google MapReduce, Hadoop) Anwendung finden. Darüber hinaus werden jedoch noch weitere Methoden zur Verfügung gestellt, wodurch Probleme, die nur schwierig in MapReduce abbildbar sind, einfach beschrieben werden können. In der Map-Phase wird eine benutzerdefinierte Funktion auf jedes Element der Liste angewendet und anschließend als Key-Value-Paar in eine neue Liste ausgegeben. In der Reduce-Phase können anschließend alle Elemente mit gleichem Key verarbeitet werden. Thrill sorgt im Hintergrund für die Verwaltung der Daten und parallele Ausführung der benutzerdefinierten Funktionen.

List Rank

Allgemein

Die Eingabe besteht aus einem unsortierten Pfad und seinem Startpunkt. Das heißt jeder Vertex kennt seinen Nachfolger, aber nicht seine Position. Ziel des Algorithmus ist den jeweiligen Rang (Position im Pfad) eines jeden Vertex zu bestimmen.

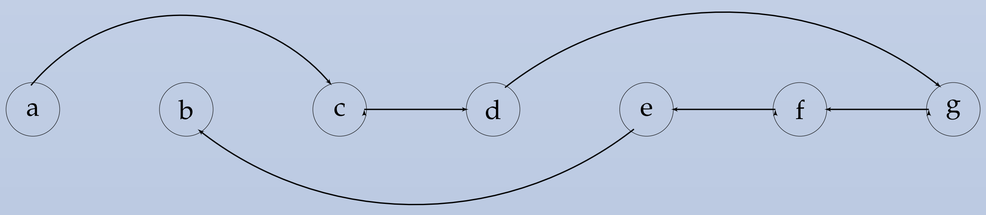


Figure 1: Eingabeliste

Die zu berechnenden Ränge sind dementsprechend:

Knoten	a	b	c	d	e	f	g
Rang	0	6	1	2	5	4	3

Table 1: Berechnete Ränge

Parallelisierung

In einem sequentiellen Algorithmus kann der Rang berechnet werden, indem ausgehend vom Startknoten sukzessiv der Nachfolger besucht wird und der Rang hierbei um eins erhöht wird. Dieser Algorithmus hat allerdings den Nachteil, dass zur Berechnung des Ranges von Knoten v der Rang des Vorgängers bekannt sein muss. Für eine parallele Lösung des Problems wird somit ein anderer Ansatz benötigt.
Unser hier gewählter Algorithmus besteht aus zwei Phasen: einer absteigenden Phase, in der die Problemgröße reduziert wird und einer aufsteigenden Phase, die ausgehend von der Lösung der Teilprobleme den Rang aller Knoten berechnet. Es wird also nach dem Divide-And-Conquer-Ansatz vorgegangen. In der absteigenden Phase müssen jeweils Knoten aus dem aktuellen Problem entfernt werden. Damit diese in der aufsteigenden Phase wieder hinzugefügt werden können, dürfen diese im Pfad jedoch nicht aufeinander folgen. Die Berechnung einer solchen Menge ist als Independent Set Problem bekannt.

Independent Set

Eine Teilmenge von Knoten ist genau dann ein Independent Set, wenn keine zwei Knoten miteinander durch eine Kante verbunden sind:

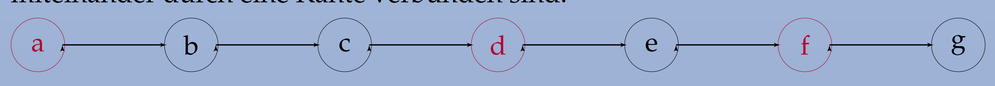


Figure 2: Beispiel independent set

Dahingegen bilden die rot markierten Knoten in der folgenden Abbildung **kein** Independent Set, da sie benachbart sind:

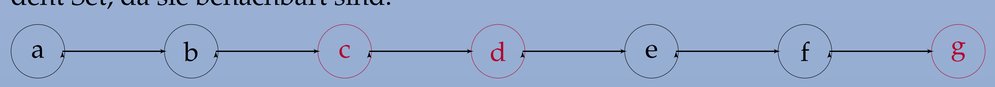
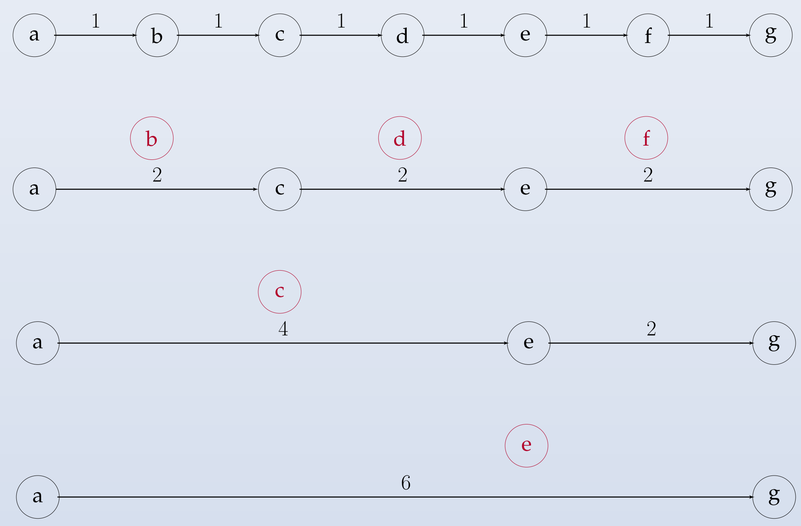


Figure 3: Beispiel kein independent set

Algorithmus

Wird ein Knoten entfernt, so werden seine beiden Kanten zusammengefasst und dabei werden die jeweiligen Gewichte addiert. Der herausgenommene Knoten merkt sich das Gewicht seiner ursprünglichen Eingangskante.



Von vorne herein bekannt ist der Rang von Knoten a, da dieser hier die Wurzel ist. Nach der absteigenden Phase ist außerdem der Rang von g bekannt.

Knoten	a	b	c	d	e	f	g
Schritt 1	0	r(a)+1	r(a) + 2	r(c) + 1	r(a)+4	r(e) + 1	r(a)+6
Schritt 2	0	1	2	r(a) + 3	4	r(a) + 5	6
Schritt 3	0	1	2	3	4	5	6

Table 2: Berechnete Ränge

Implementierung

Aufgrund des Map Reduce Konzepts von Thrill musste bei der Lösung vieler Teilprobleme von aus der Literatur bekannten Ansätze abgewichen werden. Es hat sich gezeigt, dass bei der Berechnung des Independent Sets eine approximative Lösung vorzuziehen ist. Mit unserer Implementierung, die nur einen MapReduce Schritt benötigt, konnte im Durchschnitt eine Reduktion um 25% der Knoten pro Iteration erreicht werden. Ein exakter Algorithmus würde zwischen 33% und 50% Reduktion erreichen, dabei jedoch erheblich mehr MapReduce Operationen benötigen, was im hier gegebenen Problem eine höhere Laufzeit zur Folge hätte.
Der von uns entwickelte Algorithmus zieht auf den Kanten einen boolschen Wert für den Anfangsknoten. Der Wert für den Endknoten wird anschließend auf das Komplement gesetzt. Jedem Knoten sind nun zwei boolsche Werte zugeordnet. Ein Knoten wird nur dem Independent Set hinzugefügt, falls beide seiner Werte 1 sind. Bedingt durch die Konzepte von Thrill stand uns ausschließlich MapReduce als Werkzeug zur Verfügung, um Vergleiche zwischen zwei Kanten durchzuführen. Dafür musste für jede Kante eine Hilfskante erzeugt werden, welche als übermittelte Nachricht betrachtet werden kann.
Hinsichtlich der Laufzeit des Programms konnten einige Verbesserungsmöglichkeiten identifiziert werden. Einer der Hauptpunkte war die Reduzierung von Operationen, die eine Kommunikation zwischen allen Rechenknoten zur Folge hat. So wurden auch die Berechnung des Independent Set mit anderen Berechnungen, wie das Komplement des Sets, in einem Schritt zusammengeführt. Das spart überflüssige Konkatenieren der Ergebnisse.
Eine noch größere Verbesserung konnte erzielt werden, indem die Thrill-internen Datentypen lediglich zur Berechnung herangezogen wurden und die Werte anschließend lokal zwischengespeichert wurden.

Ergebnisse

In unseren Versuchen konnte mit Hilfe des Thrill-Framework Problemgrößen bis 4 Millionen Knoten gelöst werden. Bei größere Eingaben konnte das Programm keine Lösung ermitteln, sondern lief bis zum manuellen Abbruch weiter. Der angenommene Grund hierfür ist die noch experimentelle Speicherverwaltung von Thrill. Aus diesem Grund konnten die eigentlich interessanteren Problemgrößen nicht getestet werden. Daher konnte auch bei Skalierungsexperimenten keine signifikanten Performance-Verbesserungen verzeichnet werden. Diese wäre vermutlich erst bei größeren Eingaben sichtbar.
In der folgenden Tabelle ist ein Strong-Scaling Experiment mit 1 Millionen Knoten als Eingabegraph aufgetragen.

Fazit

Anhand des List Rank Algorithmus konnte gezeigt werden, dass auch komplexe Algorithmen mit dem Konzept MapReduce und Thrill grundlegend parallel implementiert werden können. Aufgrund des experimentellen Status des Frameworks konnten keine aussagekräftigen Ergebnisse bzgl. der Performance und Skalierbarkeit ermittelt werden. Die Konzepte von Thrill konnten bei der Implementierung des Algorithmus dennoch mit der Vielzahl an nützlichen Methoden rund um den Datentyp DIA überzeugen.

References

[1] Thrill Project, `http://http://project-thrill.org/`

[2] Algorithms for memory hierarchies: advanced lectures, Ulrich et al., 2003, Springer Verlag