Universität Innsbruck

Institut für Informatik

Heuristische Optimierung der Operatorplatzierung in verteilten Stream-Verarbeitungssystemen

Cedric Immanuel Sillaber
Matrikel-Nr: 12211124

VU Einführung ins Wissenschaftliche Arbeiten

Betreuer:
Prof. Dipl.-Ing. Dr. Thomas Fahringer

May 3, 2024

Abstract

In den vergangenen Jahren wurden Big Data Applikationen stets populärer. Da die Anzahl der Daten umfangreicher wird, werden effiziente Ansätze für verteilte Stream-Datenverarbeitung (SVS) benötigt. Das Problem der Operatorplatzierung ist ein entscheidender Performancefaktor. Für das Lösen dieses Problems gibt es jedoch keine effiziente Lösung. Diese Arbeit beschäftigt sich mit einer effizienten heuristischen Methode, die versucht, die optimale Lösung zu approximieren.

1 Einführung

Im Zuge der fortschreitenden Digitalisierung entwickelte sich Datenverarbeitung zu einem zentralen Aspekt der Modernität. Der Erfolg vieler Konzerne beruht auf der Expertise, wie kontinuierliche Datenmengen effizient verarbeitet werden. Rund um die Uhr werden Daten gesammelt, die in Echtzeit verarbeitet werden müssen. SVSs wie (zitat nötig) sammeln, filtern und verarbeiten Daten. Die Daten werden von einer großen Menge an Geräten produziert. Derartige Systeme werden beispielsweise in der Analyse von Finanzmärkten [5], Verabreitung von Sozialen Netzwerk-Interaktionen und der Beobachtung von Network-Traffic [5] eingesetzt. Ein SVS besteht aus einer Menge von unabhängigen Operatoren, die eine spezifische Funktionalität ausführen. Aus der Unabhängigkeit der Operatoren lässt sich die Möglichkeit folgern, dass die Rechner im Netzwerk (Cloud-Edge [3]) lokalisiert sind, anstatt in der Cloud. In solch einem System stehen zahlreiche Ressourcen zur Verfügung. Als Operatorplatzierung bezeichnet man das Problem, die Operatoren im System optimal auf verfügbare Knoten zu platzieren. Für die Evaluation solcher Modelle werden diverse Quality-of-Service Attribute herangezogen. Dazu gehören Durchsatz, End-zu-End Latenz und Verfügbarkeit [4] [2]. Der Ansatz in dieser Arbeit versucht sich auf generalisierte QoS Attribute zu fokusieren, die einfach angepasst werden können [4]. Diese Arbeit fokusiert sich auf einen Ansatz, der bekannte Heurisiken kombiniert und optimale Lösungen effizient approximiert. Der Ansatz bezieht sich auf keine spezifische Implementierung und ist somit modell-frei. Somit kann diese Lösung für verschiedene Systeme angewendet werden.

Die optimale systematische Position in solch einem System stellt einen maßgeblichen Performancefaktor dar. Die Lösung dieses Problems ist jedoch NP-hard [2].

2 Definitionen

SVSs basieren auf einer Menge an verteilten Computer Ressourcen, die zusammen ein komplexes System ergeben. Dieser Sachverhalt kann mittels Graphentheorie beschrieben werden. Grundsätzlich gibt es zwei Abstraktionen solcher Systeme. Erstens das Datenstrom Modell und zweitens das Ressourcen Modell. Beide Systeme werden mit gerichteten, gewichteten zykelfreien Graphen $G = (V, E)$ dargestellt.

Datenstrom Modelle werden durch $G_{svs} = (V_{svs}, E_{svs})$ beschrieben. In diesem Modell beschreiben Knoten $u \in V_{svs}$ Operatoren im System. Zusätzlich sind in V_{svs} Datenquellen und Datenbecken enthalten. Man spricht hierbei von Datenbecken (vgl. "sink"), ein Punkt, an dem die Berechnungen der Operatoren zusammekommen. Zu den Operatoren gehören auch sogenannte *pinned* Operatoren [4], die Datenquellen und -becken beinhalten. Kanten $(u, v) \in E_{svs}$ beschreiben Datenstreams zwischen den Operatoren u und v. Ein Stream ist ein kontinuierliche Sequenz von Daten.

Ressourcen Modelle werden durch den Graphen $G_{res} = (V_{res}, E_{res})$ dargestellt. Dabei wird der logische Zusammenschluss zwischen verfügbaren Computing Ressourcen beschrieben. Der Knoten $u \in V_{res}$ repräsentiert solch eine Ressource. In diesem Modell beschreiben Kanten $(u, v) \in E_{res}$ eine logische Verknüpfung zwischen dem Rechnerressourcen u und v.

Für jeden Operator $i \in V_{svs}$ gibt es eine Menge an Kandidatressourcen V^i_{res} (Schreibweise wie in [4]).

Das Operator Problem bezeichnet eine Abbildung zwischen den genannten Modellen. Die Abbildung wird eingeschränkt, damit die zu minimierenden QuS Attribute eingeschränkt werden. Folglich wird der optimale Kandidat u in den Kandidatenressourcen V^i_{res} gesucht, damit Operator i auf Knoten u platziert wird. Hierbei bezieht sich das Problem auf die Inkonsistenz zwischen logisch benachbarten Operatoren im Ressourcen Modell G_{res} und optimalen Entscheidungen der Operatoren im Datenstrom Modell G_{svs}.

3 Definition von Operatorplatzierungproblem

Um das Problem formal zu definieren, verwenden wir den binären Ausdruck $x_{i,u}$ $i \in V_{svs}, u \in V_{res} : x_{i,u} = 1$ wenn Operator i auf dem Rechner u platziert wird, andernfalls $x_{i,u} = 0$

$$\sum_{i \in V_{svs}} C_i x_{i,u} < C_u \quad \forall u \in V_{res}$$

$$\sum_{u \in V_{res}^i} x_{i,u} = 1 \quad \forall i \in V_{dsp}$$

$$x_{i,u} \in \{0,1\} \quad \forall i \in V_{svs}, u \in V_{res}^i$$

Mithilfe einer *penalty function* werden die Verbindungen zwischen zwei spezifischen Knoten bezüglich der QoS Attribute bewertet. Dadruch wird ein Vergleich der Rechnerressourcen möglich. Dabei werden die Links zwischen $u \in V_{res}^i$ möglich. Auf die penalty function wird im folgenden eingegangen.

4 Penalty Function

Die Auswahl von verschiedenen Möglichkeiten $u \in V_{res}^i$ bringt Einbussen mit sich. Diverse Ressourcen haben verschiedene Lokalitäten, deren Performance durch Netzwerkdynamiken beeinflusst wird. Da die Daten übertragen werden müssen, kommen nicht vorhersehbare Network Delays dazu. Hierzu müssen Network Delay, Bandbreite und Netzwerkgeschwindigkeit[4] betrachtet werden.

4.1 Heuristiken

Wie in [2] gezeigt, ist das Operatorplatzierungproblem NP-hard. Da die initiale Platzierung der Operatoren eine tragender Faktor in der Performance einnimmt, werden effiziente Heuristiken benötigt. In dieser Abreit wird eine effiziente Methode vorgestellt, die zu einer approximierten Optimalösung führt. Dieser Ansatz beinhaltet eine Kombination mehrere bekannter Heuristiken. Ein Greedy First-Fit Ansatz in Kombination mit einer lokalen Suche findet meist lokale Optima. Um dem entgegenzuwirken wird dieser Ansatz mit einer Tabu Search verbunden. Somit werden häufiger [4] globale Optima gefunden.

Ein Greedy First-Fit Algorithmus wird für das Bin-packing Packing Problem verwenden, aber auch oft für das Operator Placement Problem [1][6]. Da diese Heuristik meist nur lokale Optima findet, werden andere Ansätze hinzugezogen. Zum einen wird Local-Search verwendet, ein Verfahren, das mit einem Greedy Ansatz über einen Teil der Funktion iteriert. Da auch dieses Verfahren dazu neigt, lokale Optima auszuwählen, wird zusätzlich Tabu Search implementiert. Die drei Heuristiken werden gekonnt kombiniert und führen somit zu einer besseren Approximation.

References

[1] Leonardo Aniello, Roberto Baldoni, and Leonardo Querzoni. Adaptive online scheduling in storm. In *Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems*, DEBS '13, page 207–218, New York, NY, USA, 2013. Association for Computing Machinery.

[2] Valeria Cardellini, Vincenzo Grassi, Francesco Lo Presti, and Matteo Nardelli. Optimal operator placement for distributed stream processing applications. In *Proceedings of the 10th ACM International Conference on Distributed and Event-Based Systems*, DEBS '16, page 69–80, New York, NY, USA, 2016. Association for Computing Machinery.

[3] Marcos Dias de Assunção, Alexandre da Silva Veith, and Rajkumar Buyya. Distributed data stream processing and edge computing: A survey on resource elasticity and future directions. *Journal of Network and Computer Applications*, 103:1–17, 2018.

[4] Matteo Nardelli, Valeria Cardellini, Vincenzo Grassi, and Francesco Lo Presti. Efficient operator placement for distributed data stream processing applications. *IEEE Transactions on Parallel and Distributed Systems*, 30(8):1753–1767, 2019.

[5] P. Pietzuch, J. Ledlie, J. Shneidman, M. Roussopoulos, M. Welsh, and M. Seltzer. Network-aware operator placement for stream-processing systems. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 49–49, 2006.

[6] Jielong Xu, Zhenhua Chen, Jian Tang, and Sen Su. T-storm: Traffic-aware online scheduling in storm. In *2014 IEEE 34th International Conference on Distributed Computing Systems*, pages 535–544, 2014.