

Heuristische Optimierung der Operatorplatzierung in verteilten Stream-Verarbeitungssystemen

Cedric Sillaber

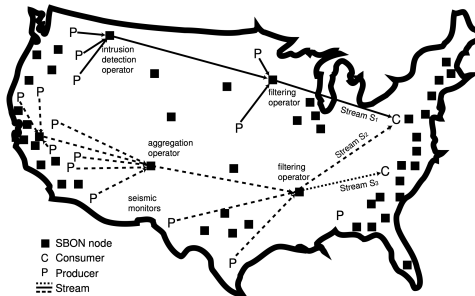
June 2024

Motivation

- Verteilte Stream-Verarbeitungssysteme bestehen aus vielen physischen Rechnern
- Welcher Rechner führt welche Operationen aus?
 - Berechnung optimaler Operatorplatzierung ist NP-schwer!
→ Approximierung durch bekannte Heuristiken
- Ziel: Latenz, Netzwerkauslastung, Verfügbarkeit, etc. optimieren

Was ist ein Operator?

- Verarbeitungseinheit in einem Stream-Verarbeitungssystem
- Beispiele: Filter, Aggregatoren, Joins



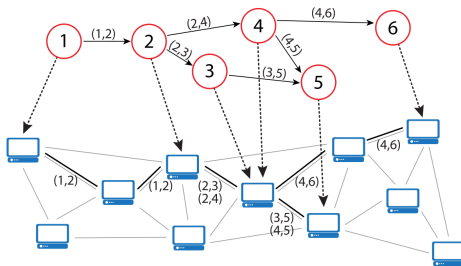
Quelle: P. Pietzuch et al., Network-aware operator placement for stream-processing systems.

Formale Definition von Streamverarbeitungssystemen

- Zwei Modelle, definiert durch gerichtete gewichtete Graphen $G = (V, E)$
- *Datenstrom-Modell* beschreibt den Fluss von Datenströmen
 - Was sind Quellen, was sind Senken? Wohin fließen Daten?
 - Wo sollte sortiert, gefiltert, gejoint werden?
 - dargestellt mit $G_{svs} = (V_{svs}, E_{svs})$
- *Ressourcen-Modell* beschreibt physische Rechner und deren Verknüpfungen
 - dargestellt mit $G_{res} = (V_{res}, E_{res})$

Problem der Operatorplatzierung

- Mapping zwischen *Datenstrom-* und *Ressourcen-Modell* (G_{sus} und G_{res})
- Möglichst effizient - Minimierung von Quality of Service Attributen



Quelle: Matteo Nardelli et al., Efficient Operator Placement for Distributed Data Stream Processing Applications

Formale Definition des OPPs

- Formal definiert mit:

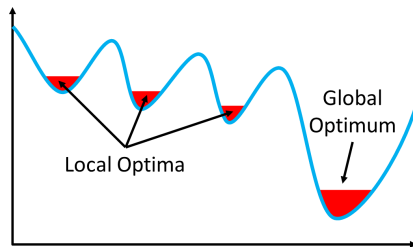
$$\begin{aligned} & \arg \min_x F(x) \\ & \sum_{i \in V_{svs}} C_i x_{i,u} < C_u \quad \forall u \in V_{res} \\ & \sum_{u \in V_{res}^i} x_{i,u} = 1 \quad \forall i \in V_{dsp} \\ & x_{i,u} \in \{0, 1\} \quad \forall i \in V_{svs}, u \in V_{res}^i \end{aligned}$$

Zu minimierende Funktion F :

$$F(x) = w_r \frac{R(x) - R_{min}}{R_{max} - R_{min}} + w_a \frac{\log A_{max} - \log A(x)}{\log A_{max} - \log A_{min}} + w_z \frac{Z(x) - Z_{min}}{Z_{max} - Z_{min}}$$

Heuristiken - Greedy First-Fit

- Näherungsverfahren zur Lösung von Optimierungsproblemen
- **Greedy First-Fit:**
 - Physische Ressourcen basierend auf Straffunktion sortiert (*greedy*)
 - Erster passender Rechner nimmt Operator auf (*first-fit*)
 - Vorteile: Einfachheit, schnelle Berechnungen
 - Nachteile: Kann in lokalen Optima steckenbleiben



Quelle: <https://www.allaboutlean.com/polca-pros-and-cons/local-global-optimum/>

Heuristiken - Lokale Suche

- Erweiterung von Greedy First-Fit
- Ziel: Verbesserung der initialen Lösung
 - Verschiebung von Operatoren zwischen Rechnern zur Optimierung
 - Vermeidung von lokalen Optima
- Vorteil: Bessere Lösung als Greedy First-Fit allein
- Nachteil: Höherer Berechnungsaufwand

Algorithm 4. Local Search

```
1: function LOCALSEARCH( $G_{dsp}, G_{res}$ )
2:   Input:  $G_{dsp}$ , DSP application graph
3:   Input:  $G_{res}$ , computing resource graph
4:    $P \leftarrow$  resources hosting the pinned operators of  $G_{dsp}$ 
5:    $L \leftarrow$  resources of  $G_{res}$  sorted by the cumulative
6:     link penalty with respect to nodes in  $P$ 
7:    $S \leftarrow$  solve GREEDYFIRST-FIT( $G_{dsp}, L$ )
8:   do ▷ local search
9:      $F \leftarrow$  value of the objective function for  $S$ 
10:     $S \leftarrow$  improve  $S$  by co-locating operators
11:     $S \leftarrow$  improve  $S$  by swapping resources
12:     $S \leftarrow$  improve  $S$  by relocating a single operator
13:     $F' \leftarrow$  value of the objective function for  $S$ 
14:    while  $F' < F$  ▷ placement solution is improved
15:      return  $S$ 
16: end function
```

Heuristiken - Tabu Suche

- Aufbauend auf Greedy und Lokaler Suche: *Tabu Suche*
 - Vorteile: Noch bessere Lösung
 - Nachteil: Noch höherer Berechnungsaufwand

Experimentelle Ergebnisse

- Vergleich der Heuristiken Greedy First-Fit, Lokaler Suche und Tabu Suche
- Metriken: Antwortzeit, Netzwerklatenz und Verfügbarkeit
- Vergleich von Laufzeit und Qualität für zwei Topologien

Vergleich der Heuristiken

Methode		SA	RA
ODP	LZ	41.4	915.2
	LZ	2174.8	32193.9
Greedy First-Fit	LZ	0.06	0.00001
	QE	7%	5%
Greedy First-Fit (keine δ)	LZ	0.057	0.00001
	QE	7%	24%
Lokale Suche	LZ	213.9	91.1
	QE	1%	4%
Tabu Suche	LZ	493.1	495.8
	QE	1%	4%

BF = Beschleunigungsfaktor, QE = Qualitätseinbuße, LZ = Laufzeit (s)

- Optimale Lösung benötigt 8h, Greedy First-Fit Bruchteil einer Sekunde.
- *Kompromiss* zwischen Qualität und Laufzeit

Fazit

- Greedy First-Fit, Lokale- und Tabu Suche bieten effiziente Lösungen für das Operatorplatzierungsproblem.
 - Greedy First-Fit: schnelle, aber minderwertige Lösungen.
 - Lokale Suche und Tabu Suche: qualitativ besser, aber zeitintensiver.
- Alle Heuristiken zeigen deutliche Verkürzung der Laufzeit

Zukunftsaussichten

- Entwicklung komplexerer Heuristiken zur besseren Approximation des OPP-Problems.
- Fokussierung auf zur Laufzeit anpassbare Heuristiken für dynamische Bedingungen.
- Analyse der Heuristiken für Real-Life Topologien.