Mapping and Scheduling HPC Applications for Optimizing I/O

Universität Hamburg

Seminar Supercomputer: Forschung und Innovation

Erik Kersten - 13.12.2022

Agenda

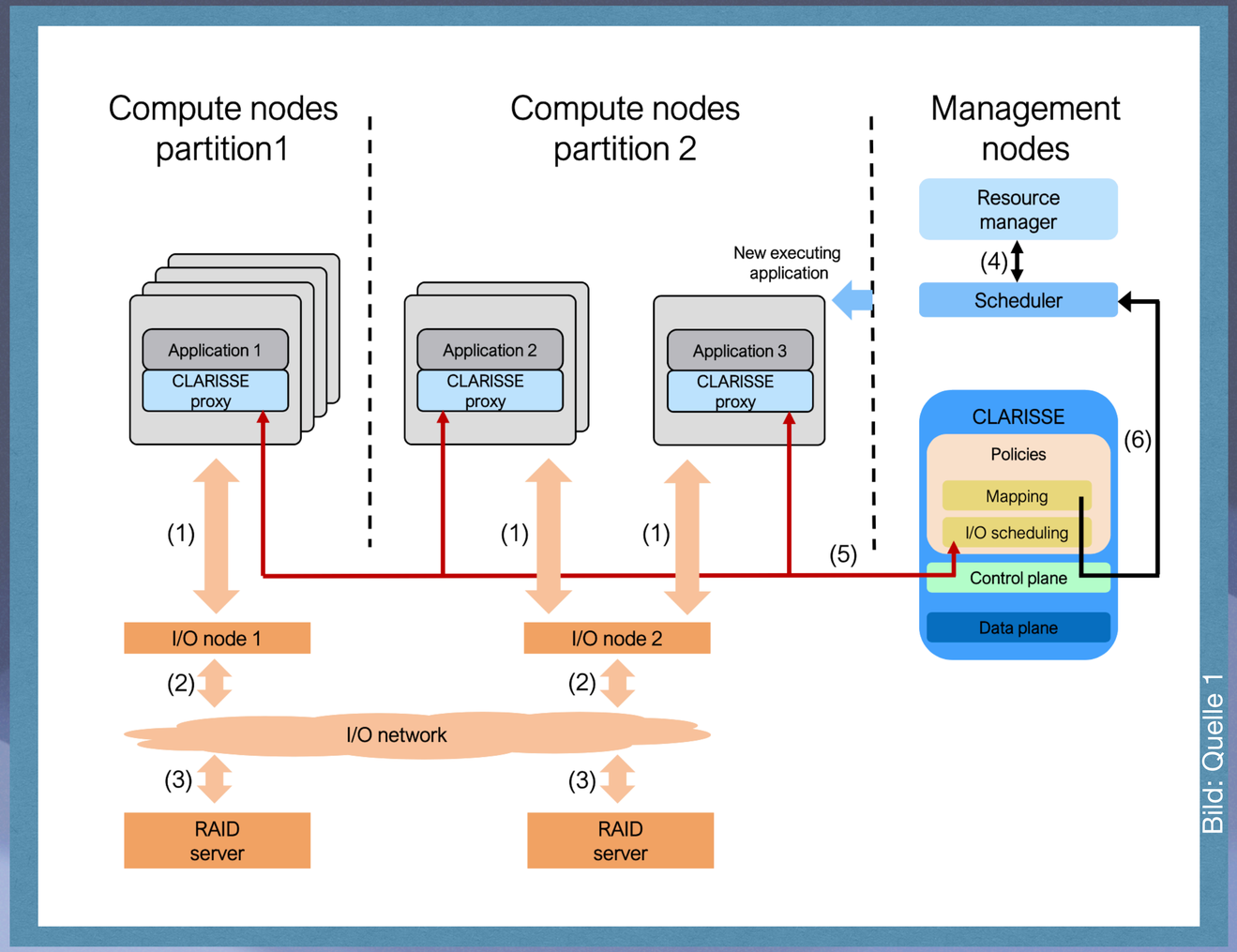
- Einleitung High-performance computing (HPC)
- Das Problem des Dateisystems
- Modell
- Pack Scheduling
  - I/P policies
  - Pack-partitioning algorithm
- Auswertung der Experimente
- Zusammenfassung

Einleitung:

High-performance Computing - I/O

- Programme teilen sich das gleiche Dateisystem
  - kann zu Konflikten führen
- I/O als Flaschenhals (Bottleneck)

Das Problem des Dateisystems



Das Problem des Dateisystems

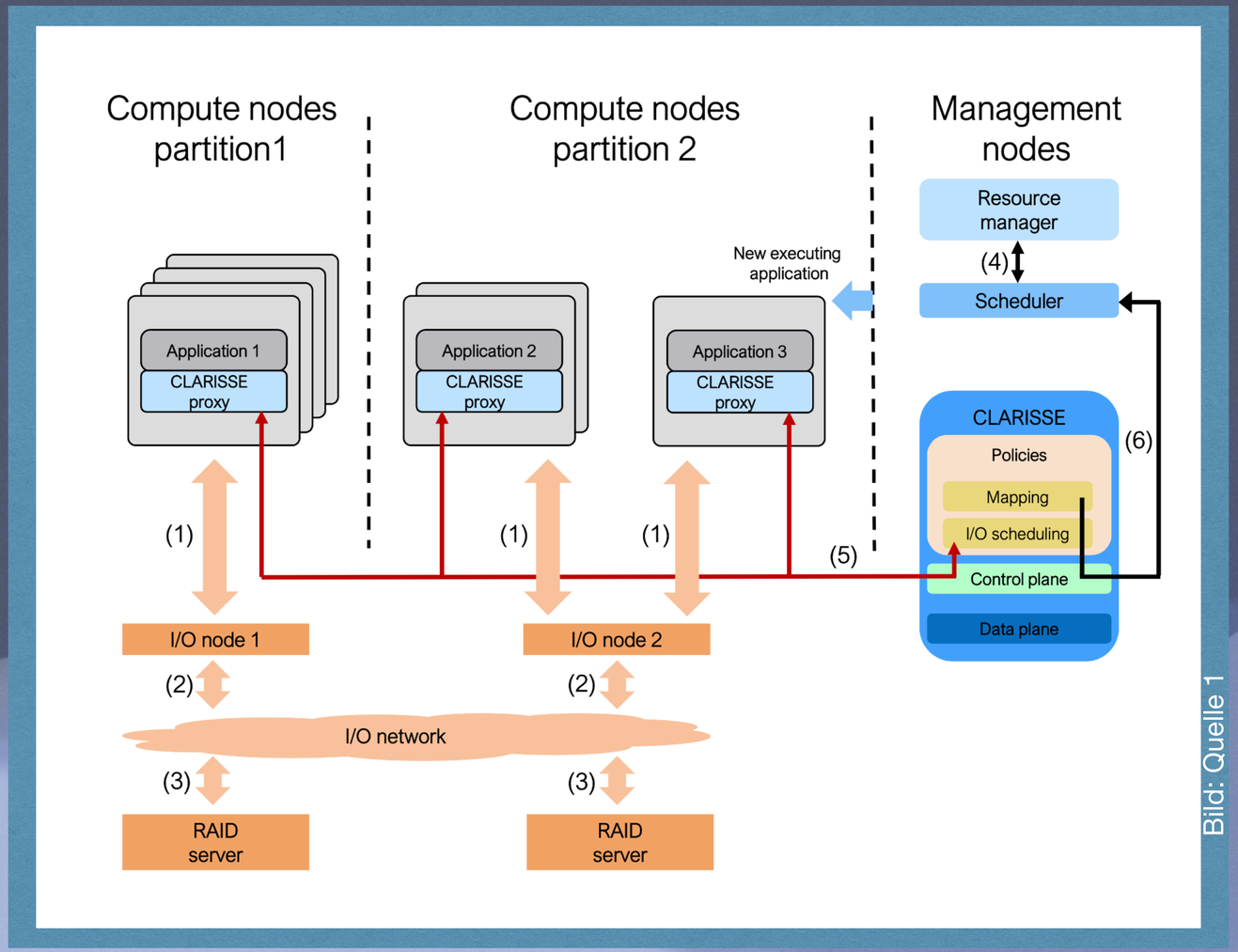


Bild: Quelle 1

<u>Das „mapping problem“:</u>
Mehrere Anwendungen wollen die gleichen I/O Ressourcen benutzen

<u>Das „scheduling problem“:</u>
In welcher Reihenfolge kann Anwendungen Zugriff auf I/O Ressourcen gegeben werden, sodass dessen Ausführungszeit minimal ist.

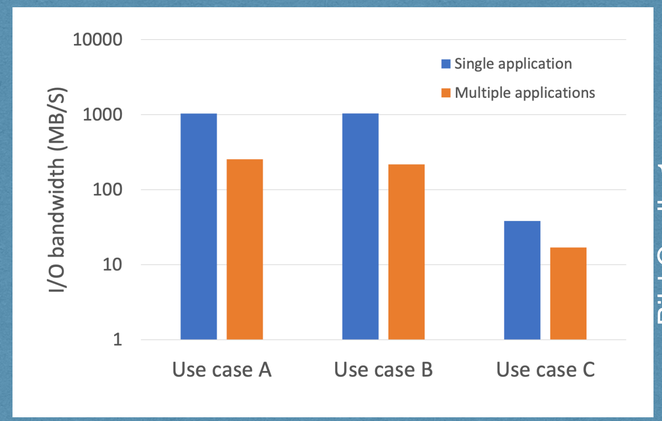


Bild Quelle:1

Das Paper

Mapping and Scheduling HPC Applications for Optimizing I/O

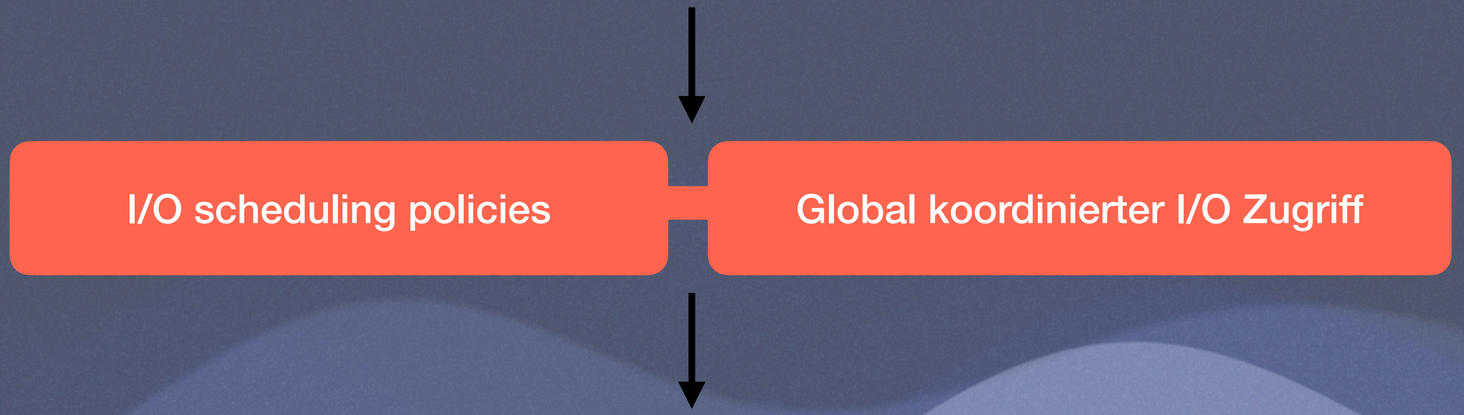
- Jesus Carretero et al.
  - Universität Carlos III - Madrid
- 30. April 2020

- Unterstützt von der „French National Research Agency (ANR)

Das Problem des Dateisystems

Ziel des Arbeit:

Ziel: Keine Verzögerung durch wartende Anwendungen



Weniger I/O Interferenzen

Modell

Grafik der Architektur

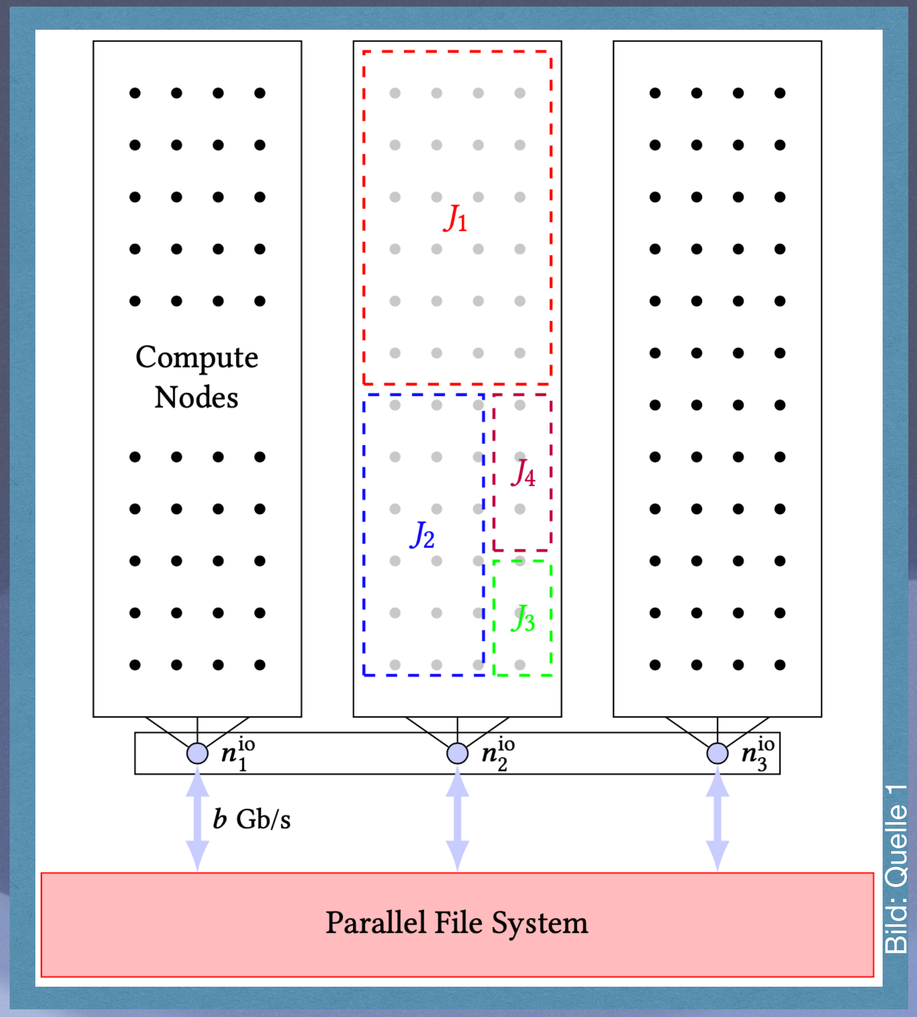


Bild: Quelle 1

Modell

Grafik der Architektur

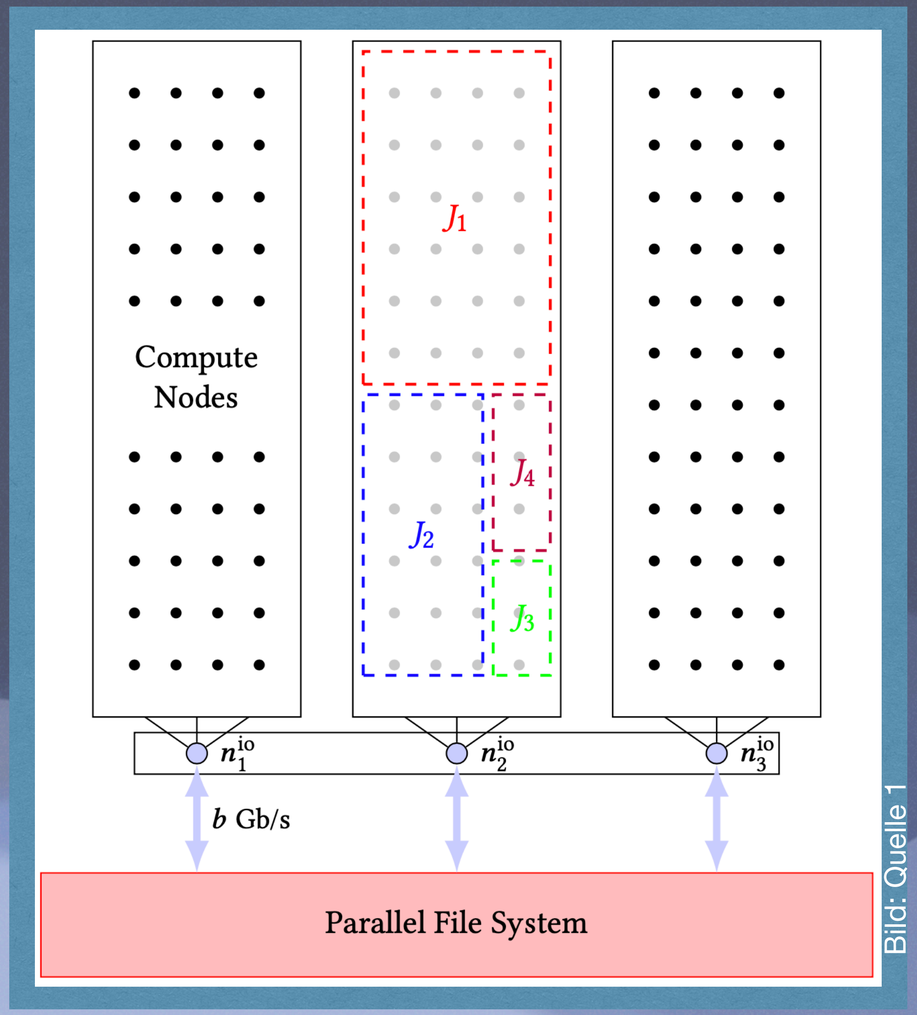


Bild: Quelle 1

- P - **Rechenknoten**
- R - **I/O Knoten**
  - jeder mit einer **Bandbreite** b

Menge an Rechenknoten: $\sum_{j=1}^{R} P_j$ $j \in \{1, \ldots, R\}$

Bandbreite b bei allen **I/O Knoten** identisch

alle R **I/O Knoten** gleich viele P **Rechenknoten**

Modell

Grafik der Architektur

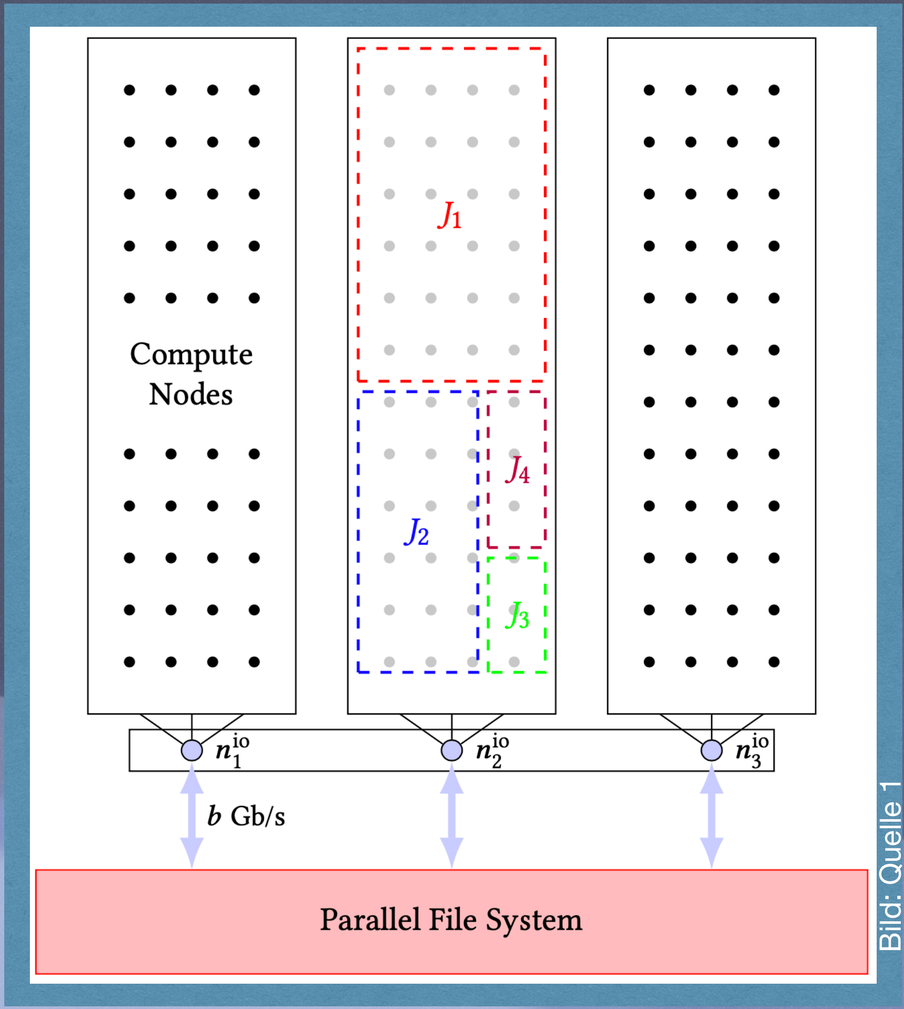


Bild: Quelle 1

- P - **Rechenknoten**
- R - **I/O Knoten**
  - jeder mit einer **Bandbreite** b

Menge an Rechenknoten: $\sum_{j=1}^{R} P_j$ $j \in \{1, \ldots, R\}$

Bandbreite b bei allen **I/O Knoten** identisch

alle R **I/O Knoten** gleich viele P **Rechenknoten**

n **Jobs** $\{J_1,\ldots,J_n\}$ mit Q_i Rechenknoten

$W_{i,j}$ - Zeit einer Rechenoperation

$V_{i,j}$ - Menge an Daten für I/O Operationen

Modell

Formeln

- **Zeit** T_i benötigt für Ausführung von J_i: $T_i(b) = \sum_{j \leq n_i} w_{i,j} + \frac{v_{i,j}}{b}$
- **Stretch** p_i von J_i: $p_i = \frac{\sum_{j \leq n_i} w_{i,j} + \frac{v_{i,j}}{b}}{C_i - r_i}$

r_i = Startzeit von Job i
C_i = Abschlusszeit von Job i

- Makespan C_{max} (nach Ausführung aller Jobs): $C_{max} = maxC_i$

„[…] mit einer festen Menge an Jobs J_1,…, J_n und einer Plattform mit R I/O Knoten, jeder mit einer Bandbreite b […], und verbunden mit P Rechenknoten. Finde eine Programm, welches entweder die ***makespan*** minimiert, oder den maximalen ***strech***($max_i\ p_i$) minimiert.[…]„

Übersetzt aus Quelle 1

Pack Scheduling

Mapping von 11 Anwendungen in Packs

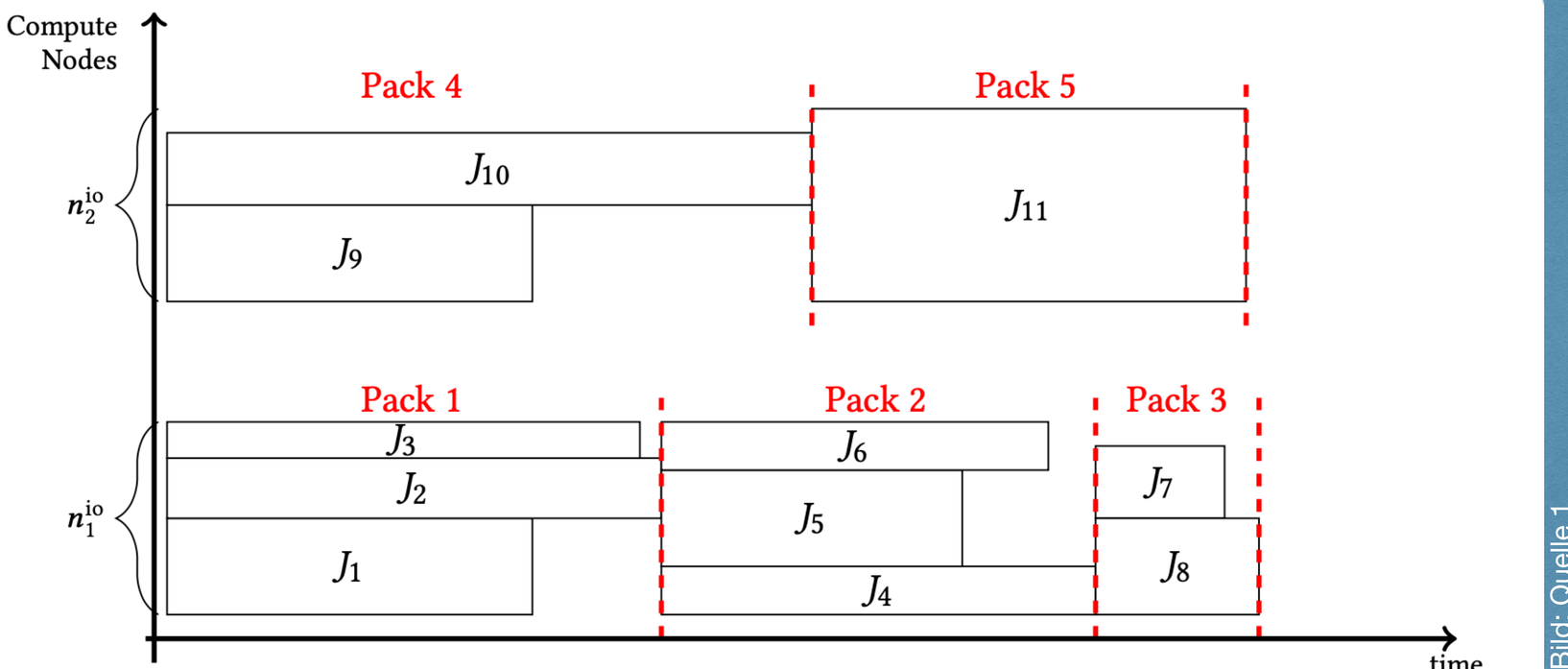


Bild: Quelle 1

Pack Scheduling

Scheduling von I/O Operationen (Pack 2)

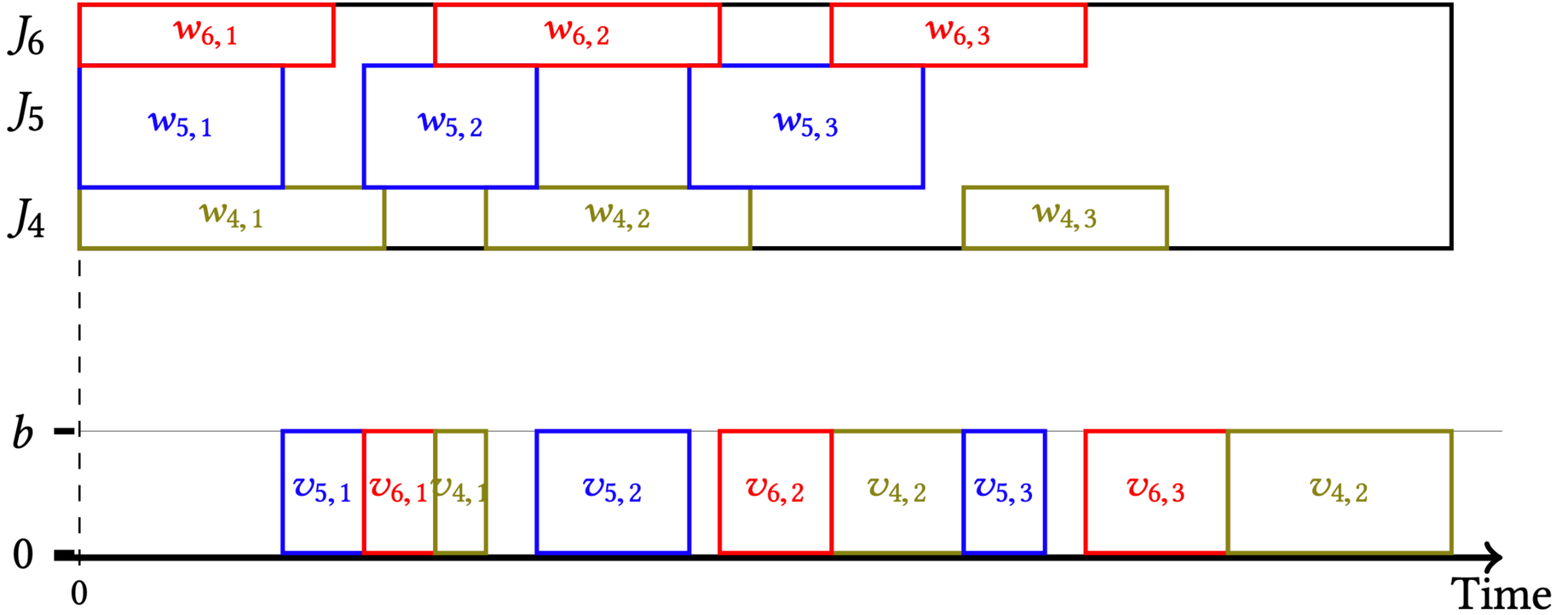


Bild: Quelle 1

Pack Scheduling

I/O Policies

1. **Lowest ID**
2. **Longest I/O** phases
3. **Shortest I/O** phases
4. **Shortest remaining** time
5. **Longest remaining** time
6. **FIFO** (first in first out)
7. **Bandwidth oriented**
8. **Stretch oriented**
   niedrigster stretch erst

Algorithm 1: Pack-partitioning algorithm.

procedure Make-Pack($J_1, \ldots, J_n$,S)
begin
Assume the jobs are sorted in decreasing order of $T_i(b) = \sum_{j \le n_i} w_{i,j} + \frac{v_{i,j}}{b}$;
Let S_P be a set of packs sorted by decreasing value of P^{Pack} (empty initially);
for $i = 1$ *to* n **do**
Find the first Pack in S_P such that

$$\sum_{j \le n_i} v_{i,j} \le (S - L^{\text{Pack}}) \cdot bT^{\text{Pack}} \tag{4}$$

$$Q_i \le P - P^{\text{Pack}} \tag{5}$$

Fit J_i in this pack;
If there is no such pack, create an empty pack;
end
return S_P
end

Quelle 1

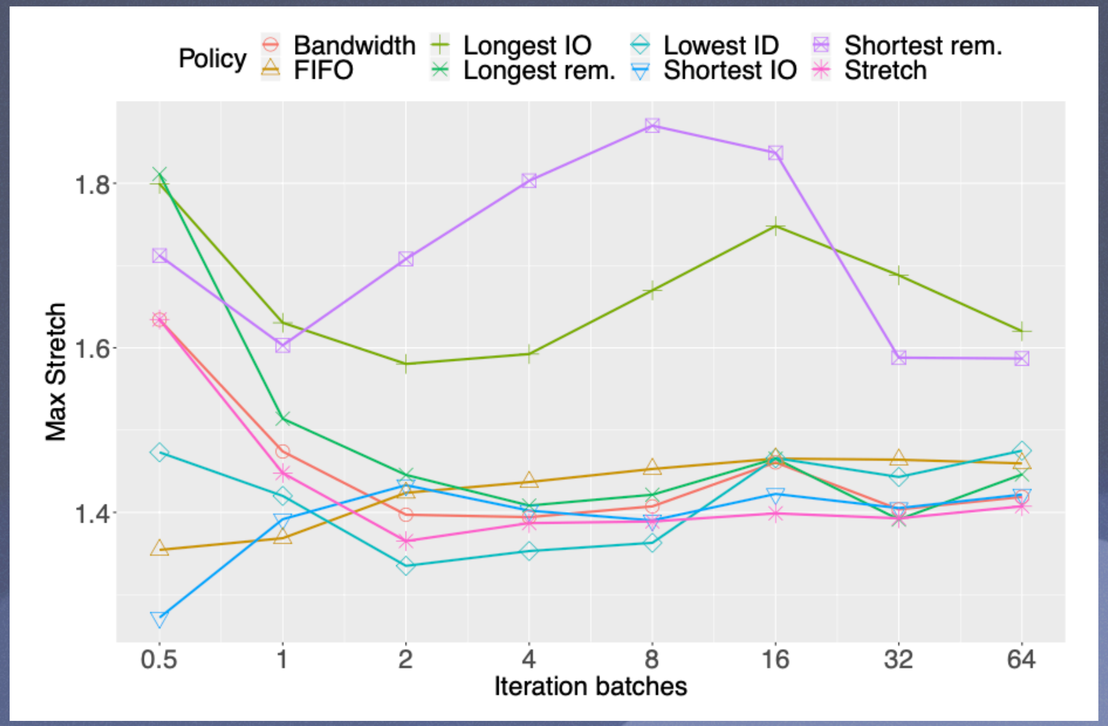
Auswertung der Ergebnisse

Application id	Matrix size	CPU duration (s)	I/O duration (s)	Iterations
1	5000	32	10	250
2	5000	16	5	500
3	5000	8	2.5	1000
4	5000	8	2.5	1000
5	5000	16	5	500

Quelle 1

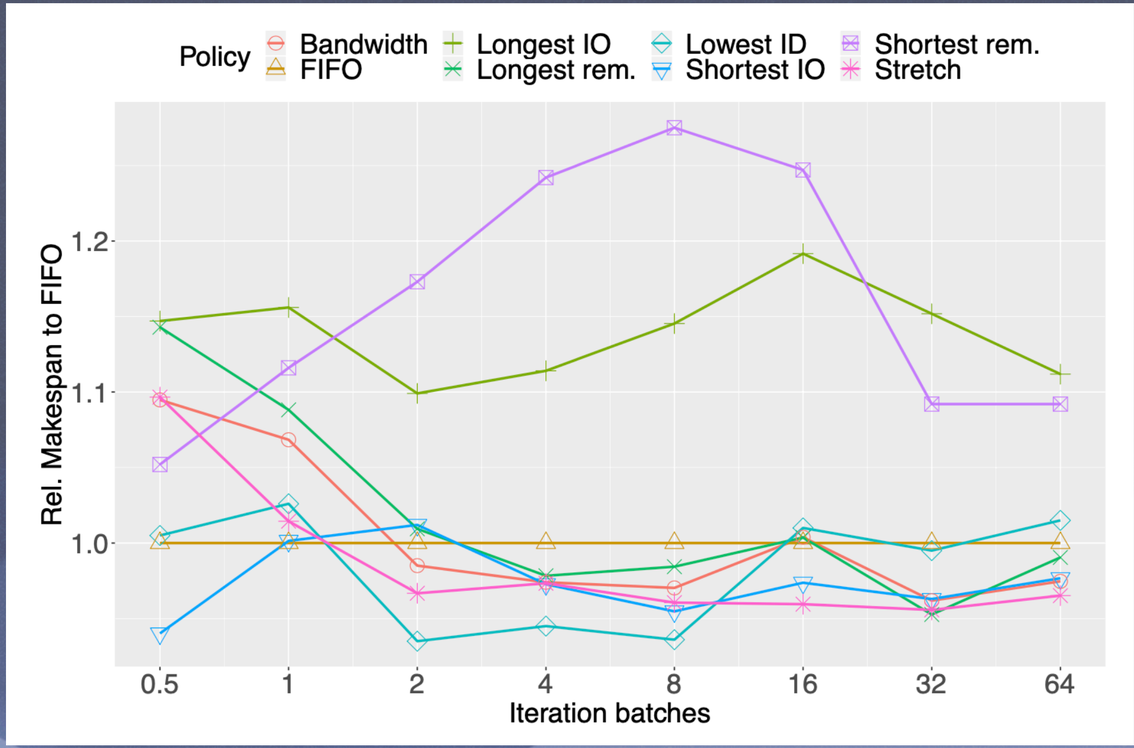
Bei einzelner Ausführung: alle gleiche Dauer

Auswertung der Ergebnisse



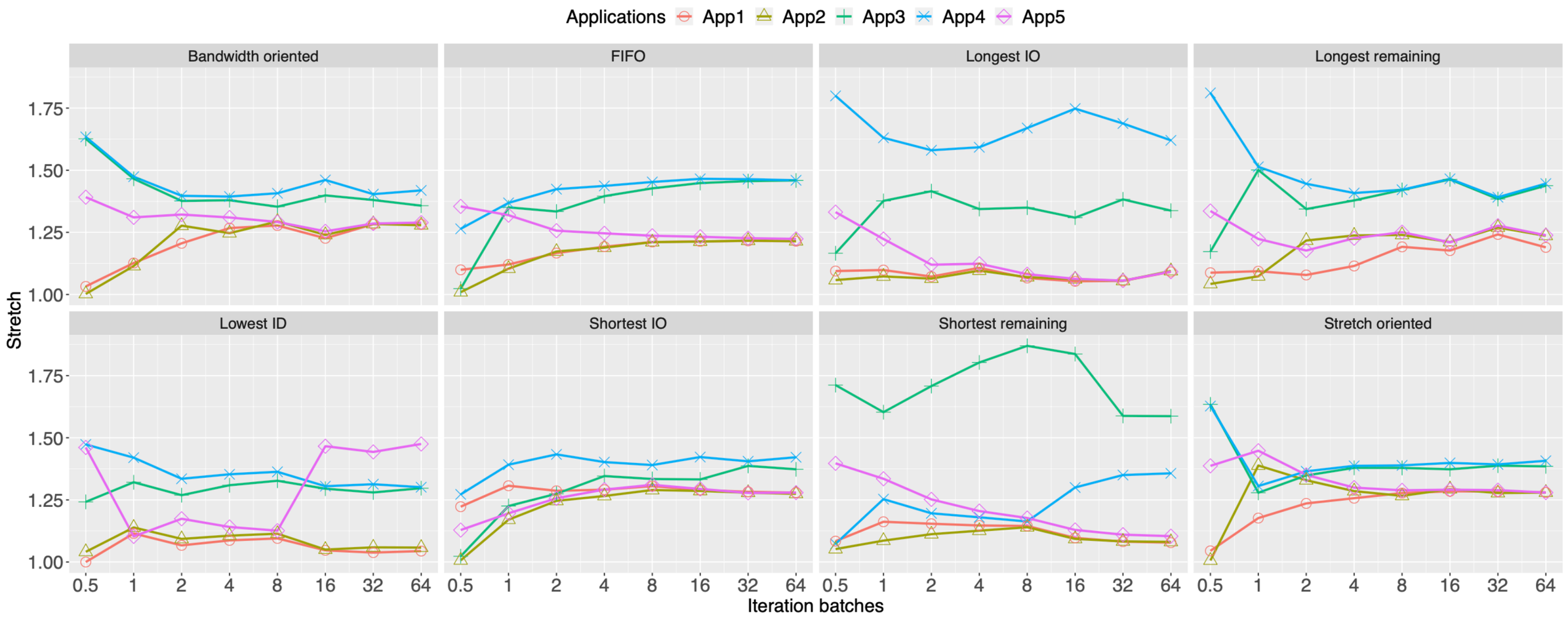
Max Stretch of the different policies when varying the number of iteration batches. (Quelle 1)

Auswertung der Ergebnisse



Relative Makespan to FIFO of the differen policies when varying the number of iteration batches. (Quelle 1)

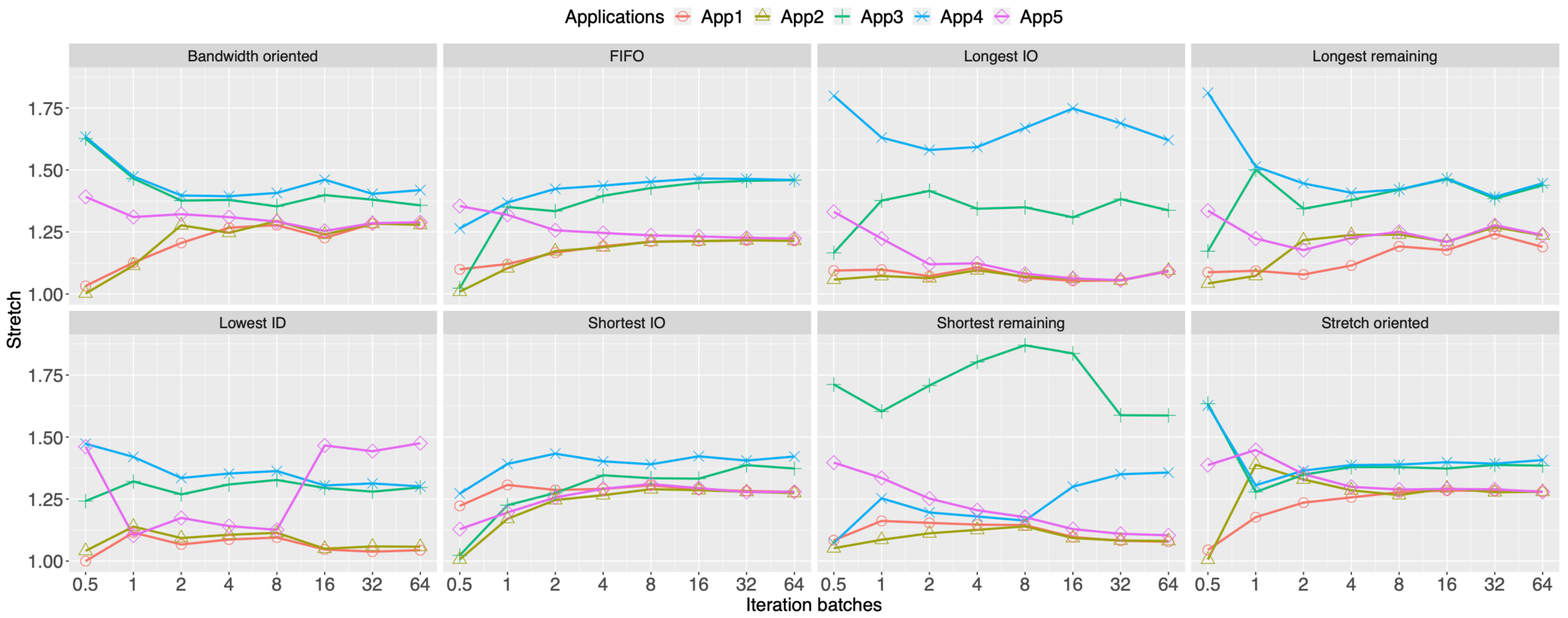
Auswertung der Ergebnisse



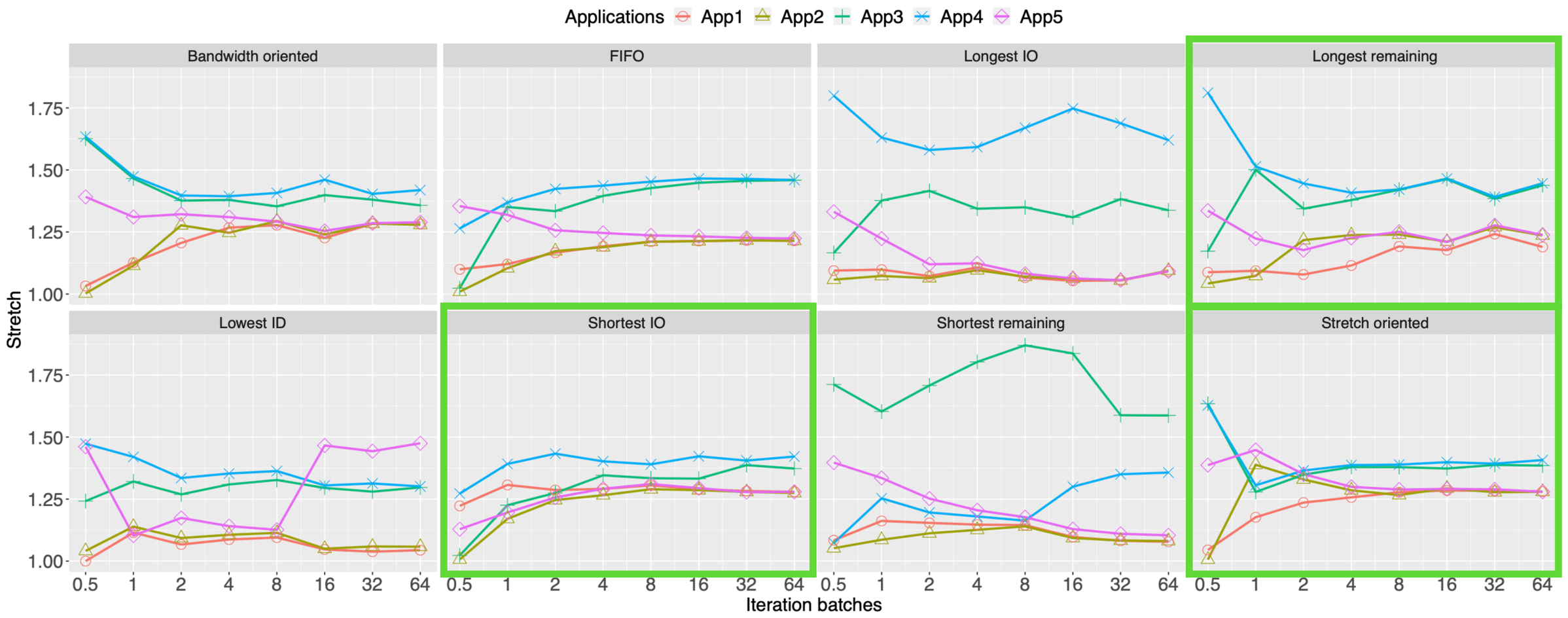
Quelle 1

Auswertun

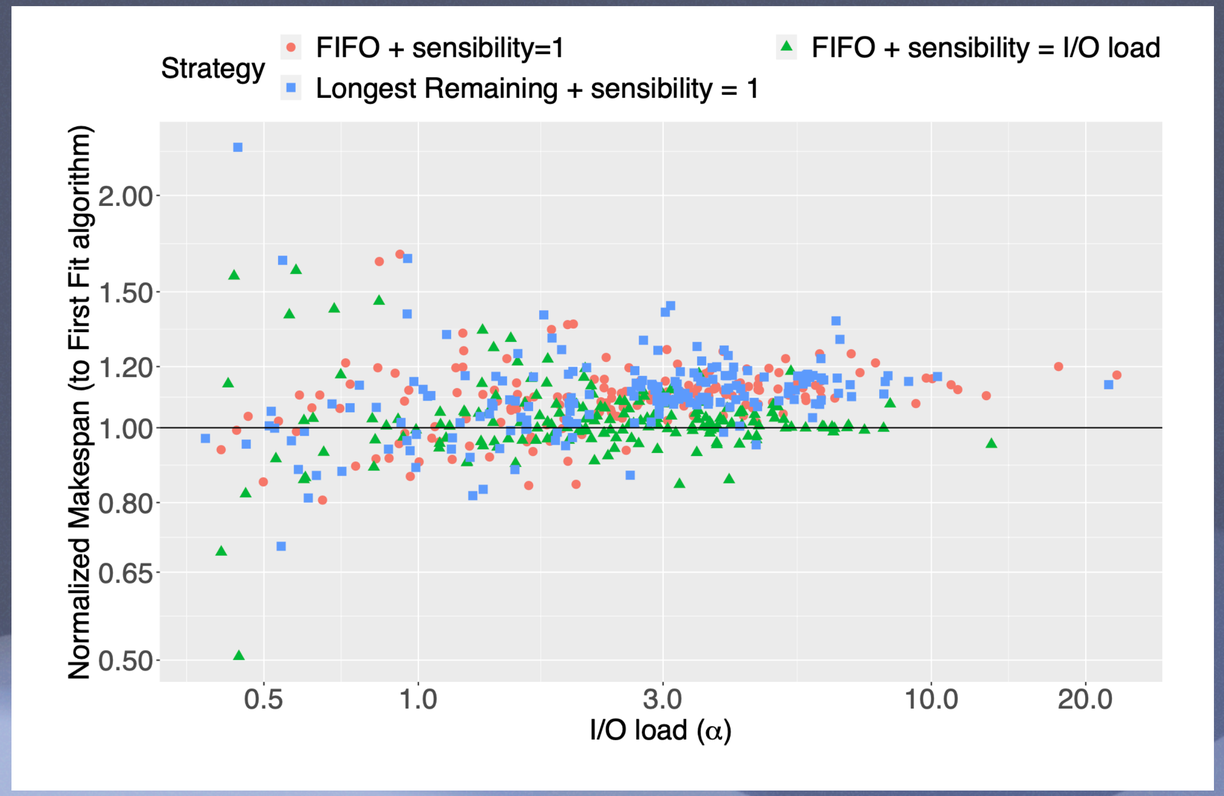
Application id	Matrix size	CPU duration (s)	I/O duration (s)	Iterations
1	5000	32	10	250
2	5000	16	5	500
3	5000	8	2.5	1000
4	5000	8	2.5	1000
5	5000	16	5	500



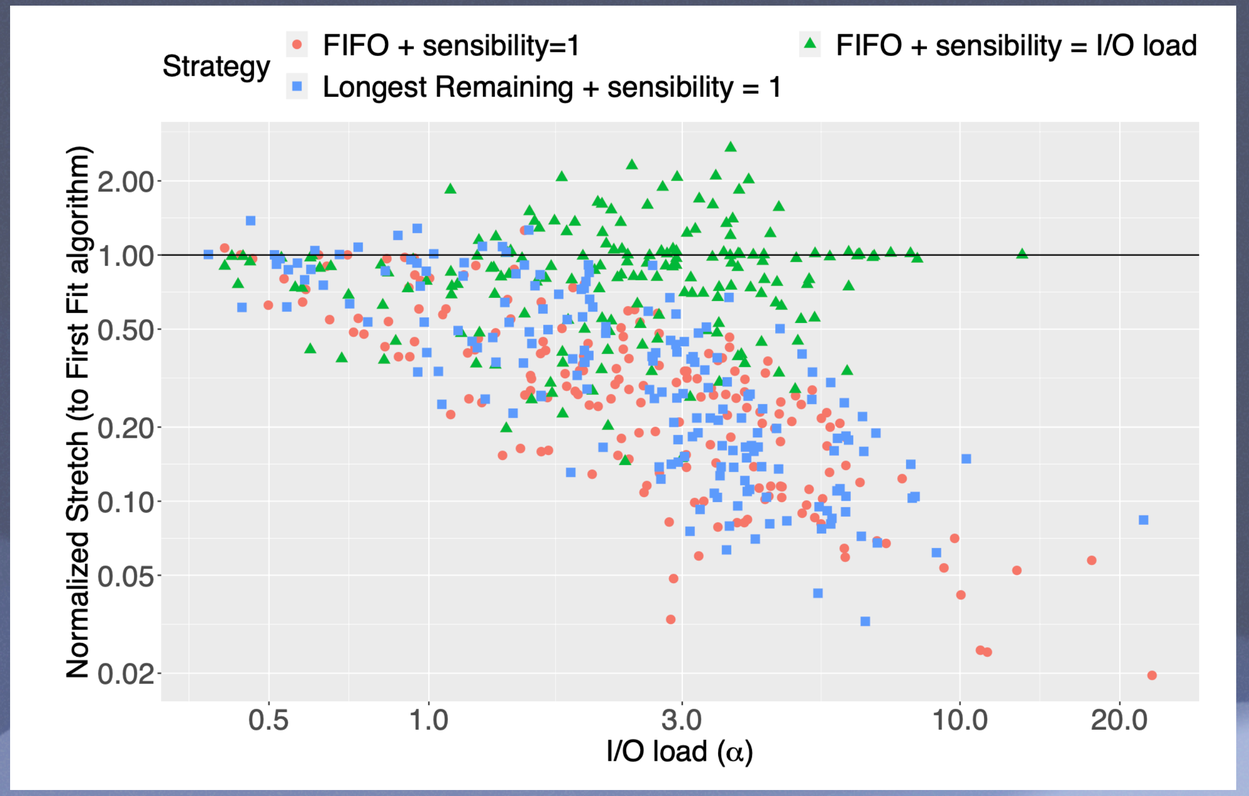
Auswertung der Ergebnisse



Auswertung der Ergebnisse



Comparison of makespan for different strategies (Quelle 1)



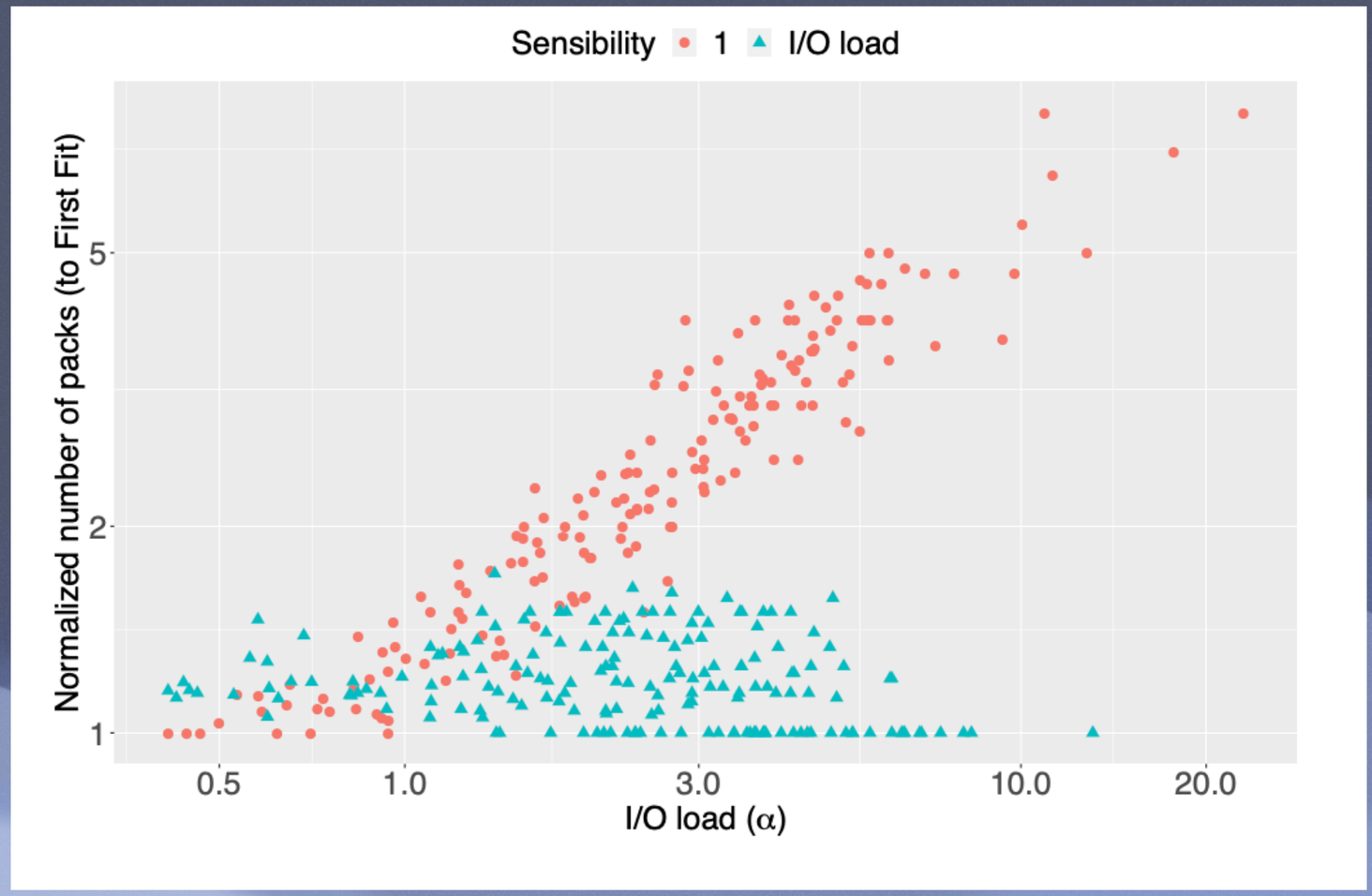
Comparison of stretch for different strategies (Quelle 1)

Auswertung der Ergebnisse

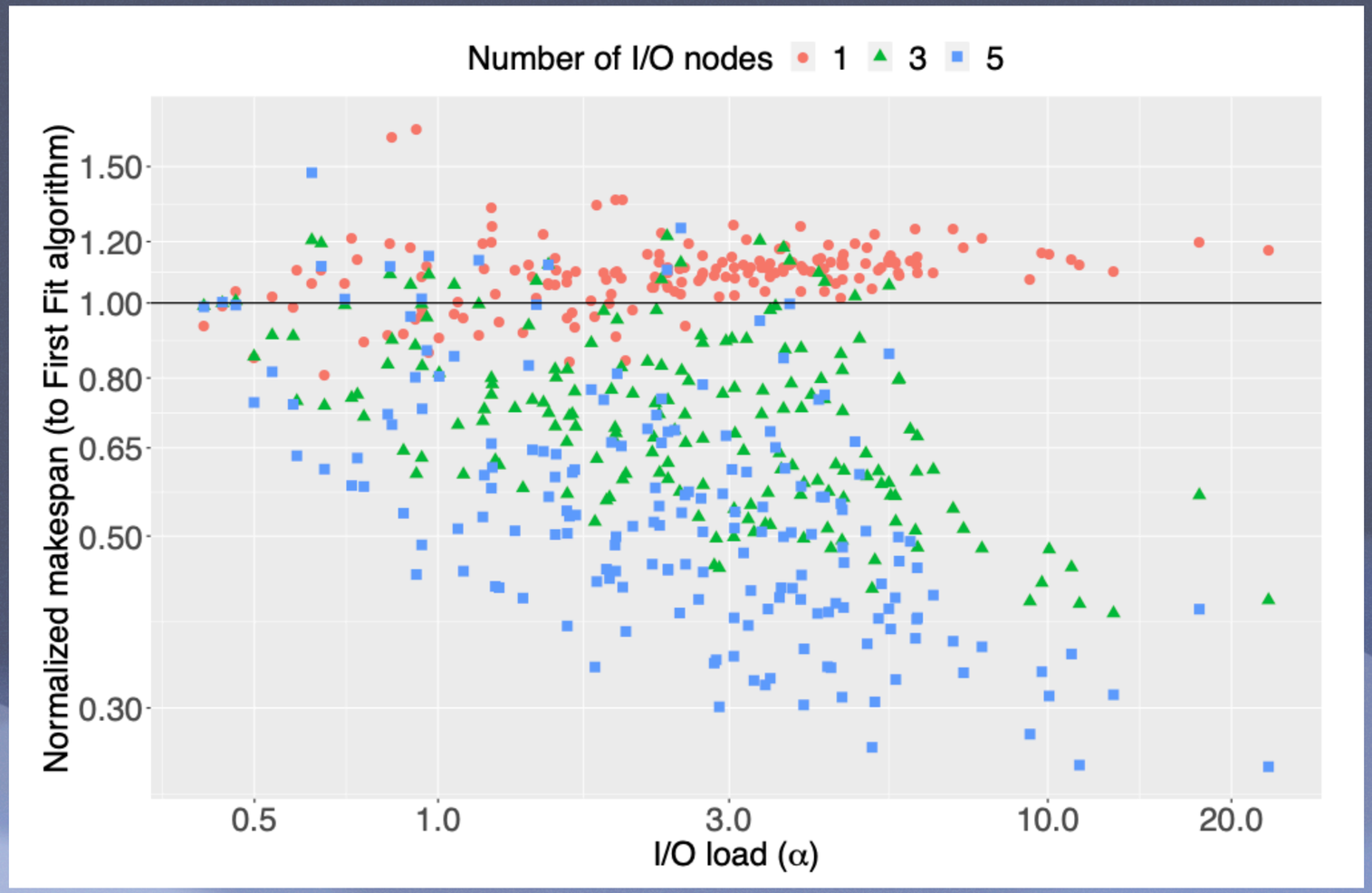
I/O Load

$$a = \frac{P \cdot \sum_{i=1}^{n} \sum_{j \leq n_i} w_{i,j} + \frac{v_{i,j}}{b}}{\sum_{i=1}^{n} (Q_i * T_i(b))}$$

Auswertung der Ergebnisse



Auswertung der Ergebnisse



Comparison of the relative makespan of Pack Par- titioning Algorithm (with sensibility =1) to the First-Fit algo- rithm with multiples I/O nodes. (Quelle 1)

Zusammenfassung

- Bei einem einzelnen I/O Knoten:
  - maximaler *stretch* verringert und niedrigere *makespan*
- Bei mehreren I/O Knoten:
  - Stärkere verbesserung von *stretch* und *makespan*

Quellen

1. Carretero, Jesus, et al. "Mapping and scheduling HPC applications for optimizing I/O." *Proceedings of the 34th ACM International Conference on Supercomputing*. 2020.

2. https://de.wikipedia.org/wiki/Universität_Carlos_III

3. F. Isaila, J. Carretero and R. Ross, "CLARISSE: A Middleware for Data-Staging Coordination and Control on Large-Scale HPC Platforms," 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2016, pp. 346-355, doi: 10.1109/CCGrid.2016.24.