Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Apriori Algortihmus

Endpräsentation

Stefan George, Felix Leupold

Gliederung

- Wiederholung Apriori
- Erweiterung: Parallelisierung
  - Parallele Programmierung in Python
  - Parallelisierungszenarien
  - Implementierung
  - Ergebnisse/Benchmarks
- Usecase
  - Lift
  - Conviction
  - Ergebnisse

Wiederholung Apriori

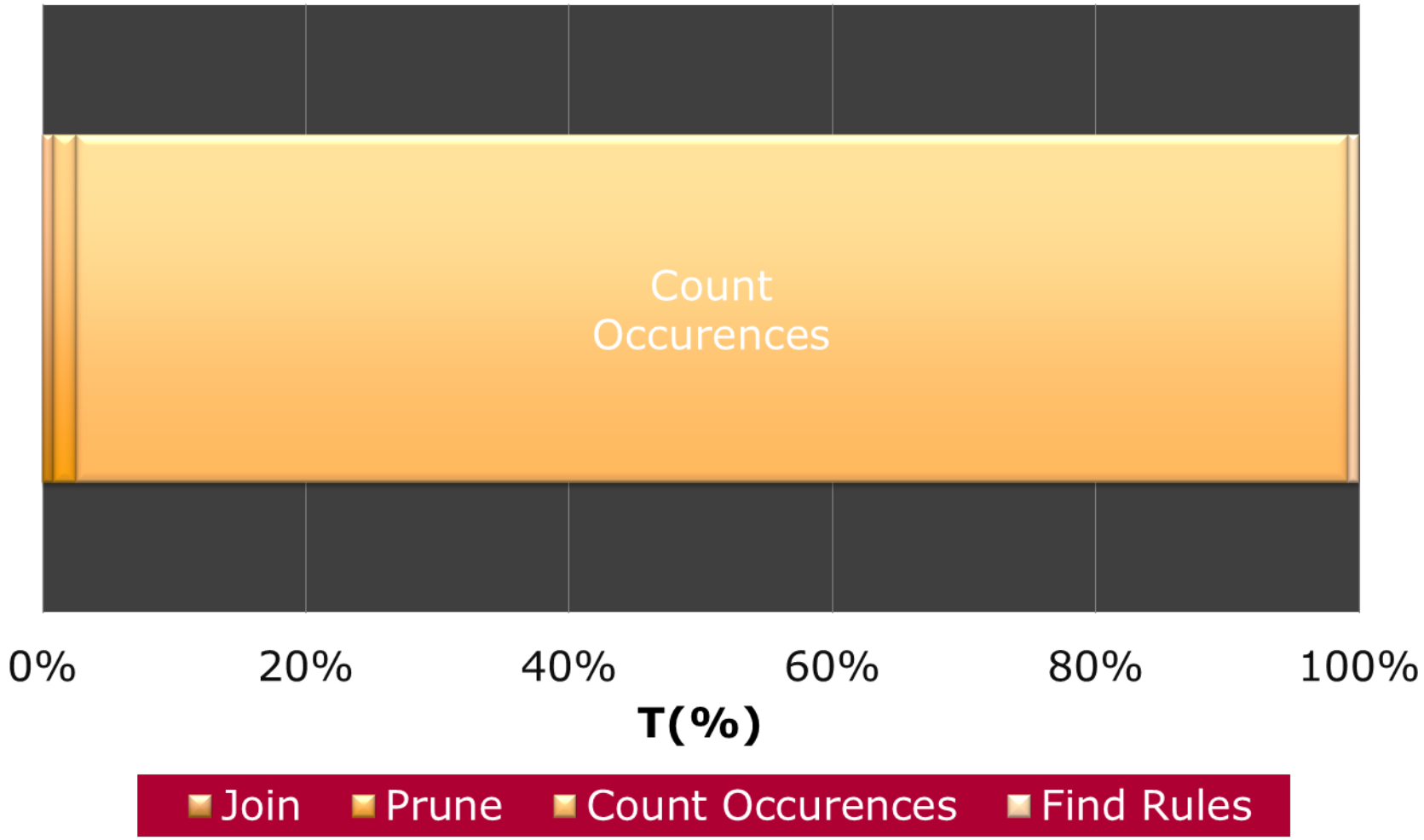
- Algorithmus zum Auffinden von Assoziationsregeln
- Wichtige Parameter: Support und Confidence
- Apriori Überlegung:

Wenn eine Teilmenge einer Menge M klein ist, dann ist die Menge M auch auf jeden Fall klein

- Kandidatengenerierung iterativ: Join und Prune
- Regeln aus Itemsets mit ausreichend Support generieren

$$A \rightarrow (L - A) \quad \leftrightarrow \quad \frac{\sup(L)}{\sup(A)} > \min Conf$$

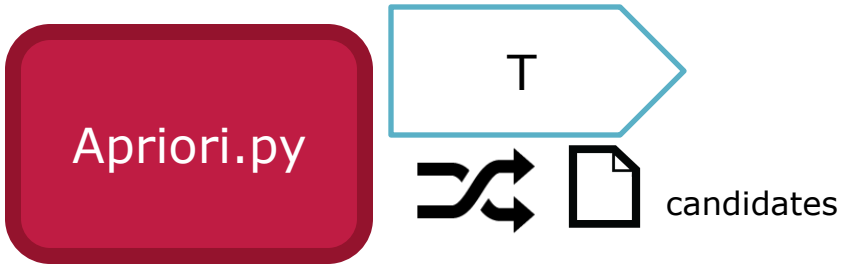
Motivation für Parallelisierung



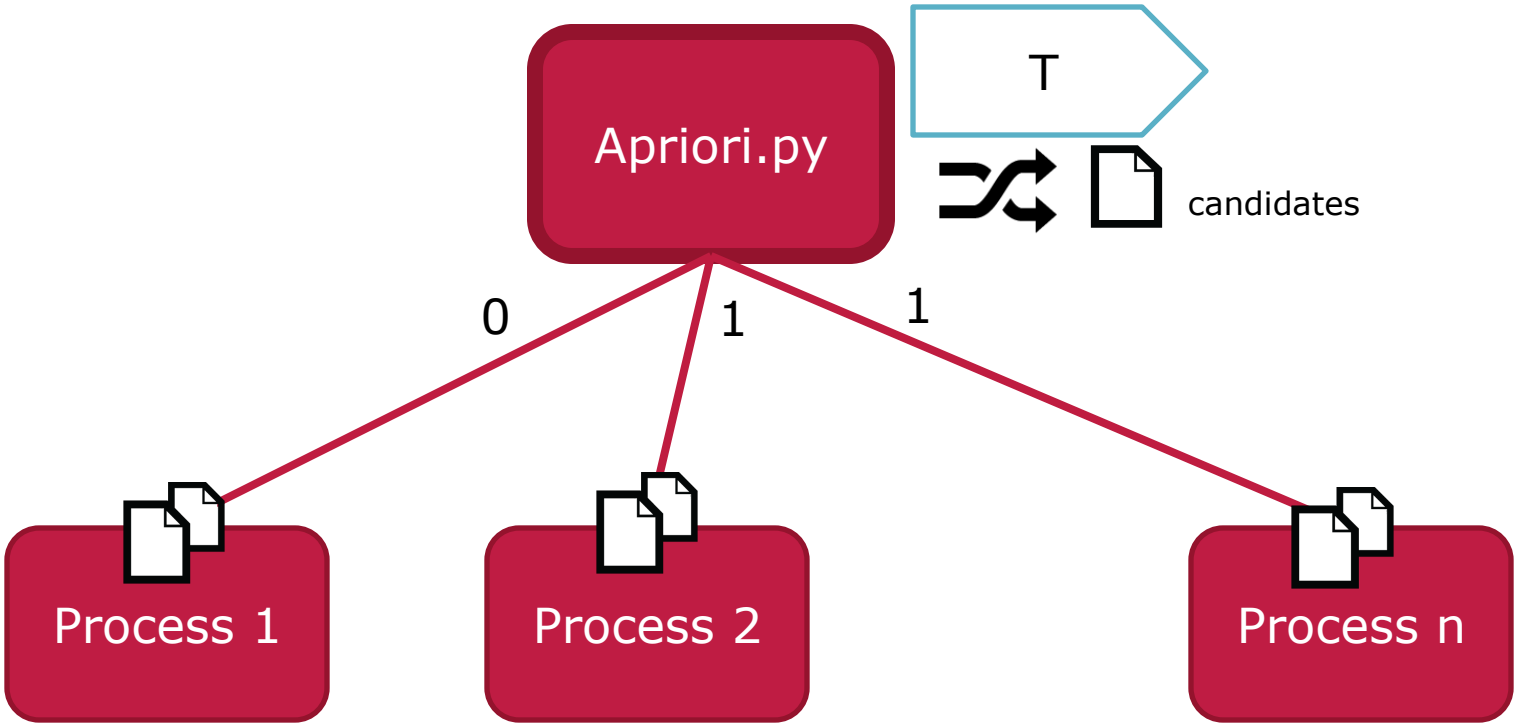
Parallelisierung in Python

- Multiprocessing vs. Multithreading
- Threads haben „shared memory“
- Deutlich mehr overhead bei Context Switch zwischen Prozessen
- Global Interpreter Lock in Python
  - Nur ein Thread kann gleichzeitig Bytecode ausführen
  - Keine Parallelisierung bei CPU lastigem Code
- Für uns kommt nur Multiprocessing in Frage

Parallelisierungsszenarien (1/3)



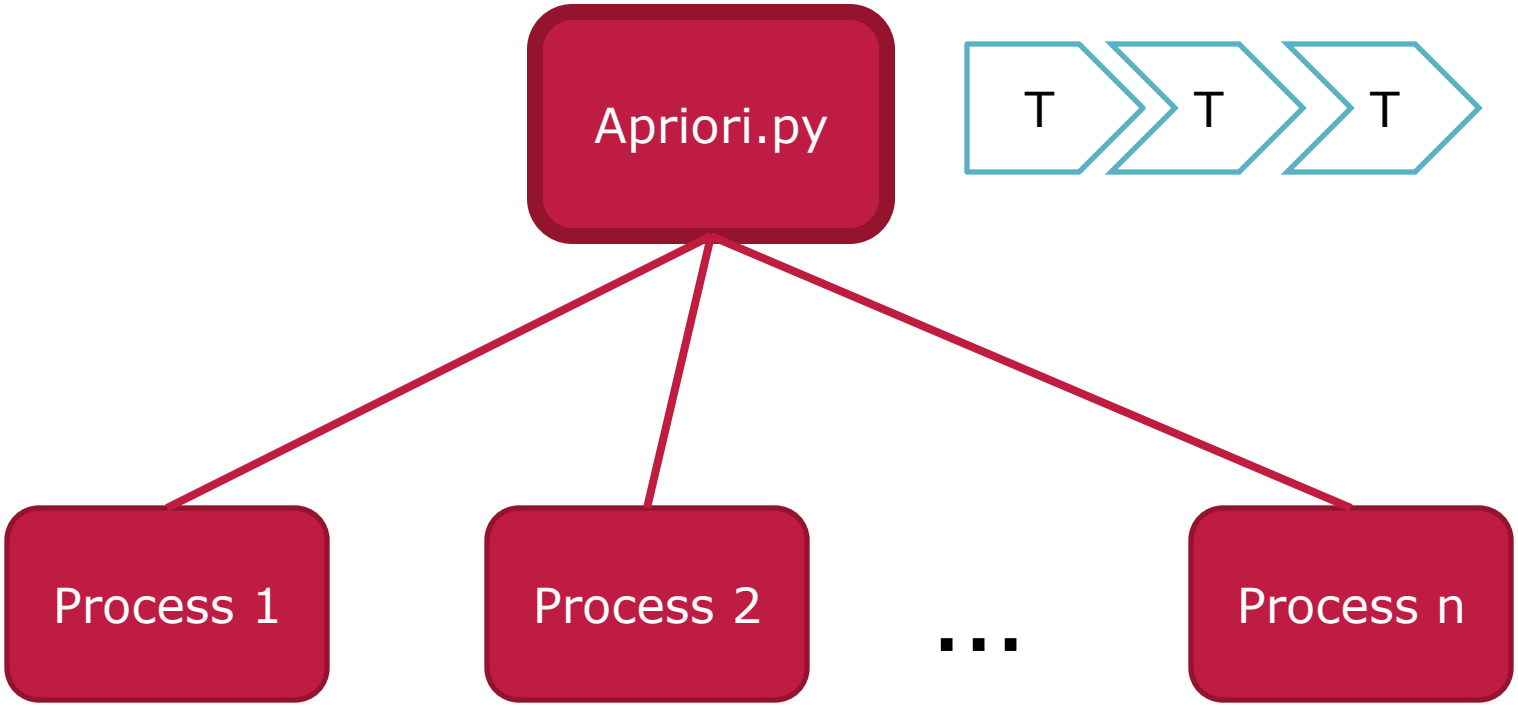
Parallelisierungsszenarien (1/3)



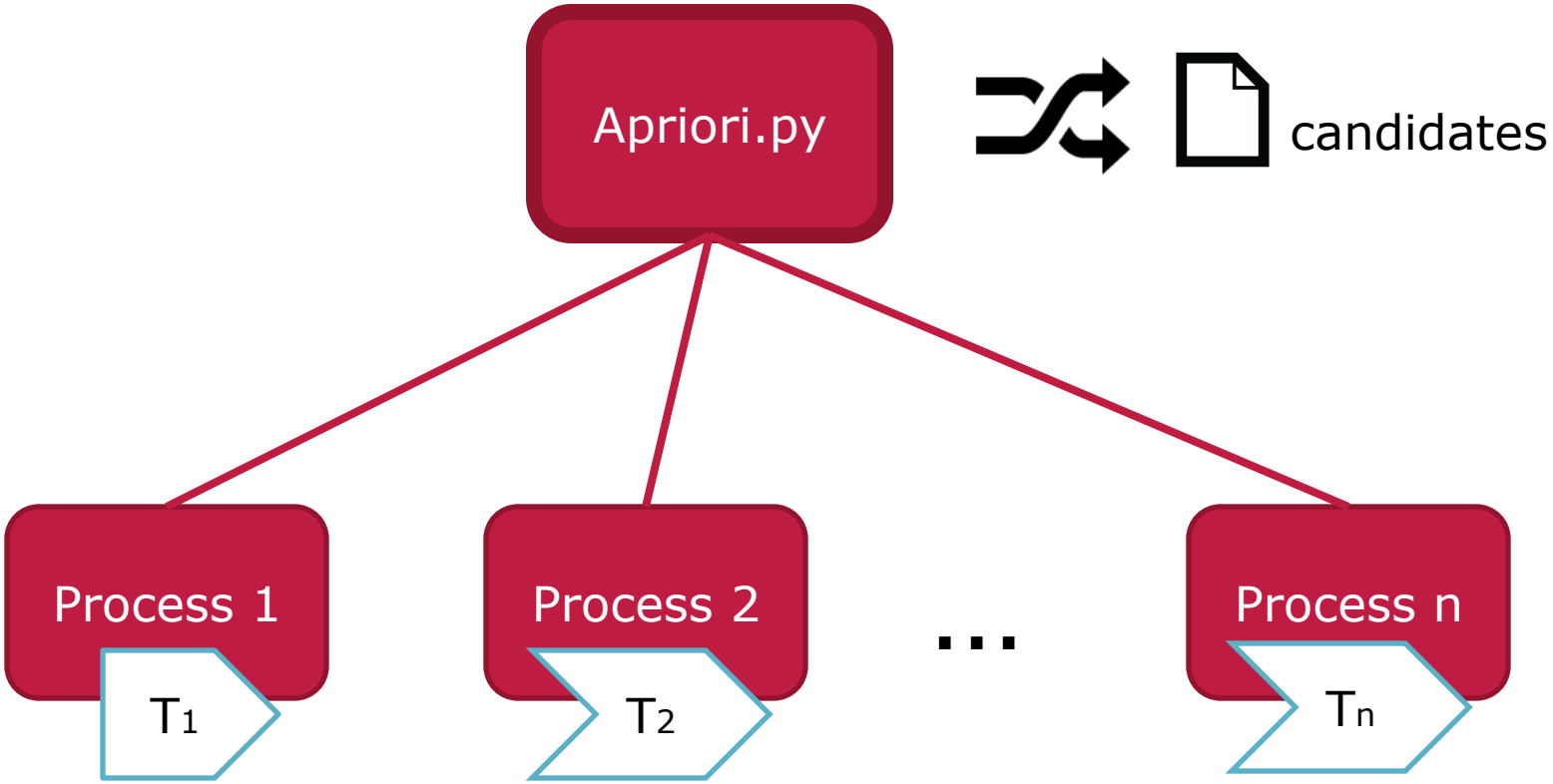
Analyse (1/3)

- Kein Initialisierungsaufwand — 0
- Datenübertragung in Schritt k: |D| * $|C_k|$ — 5Mio * 400
- Ergebnisgröße: $|C_k|$ — 400
- Gesamtkosten = $\sum_k (|D|+1) * |C_k|$ — 2Mrd
- Parallelisierung muss grob granularer sein.

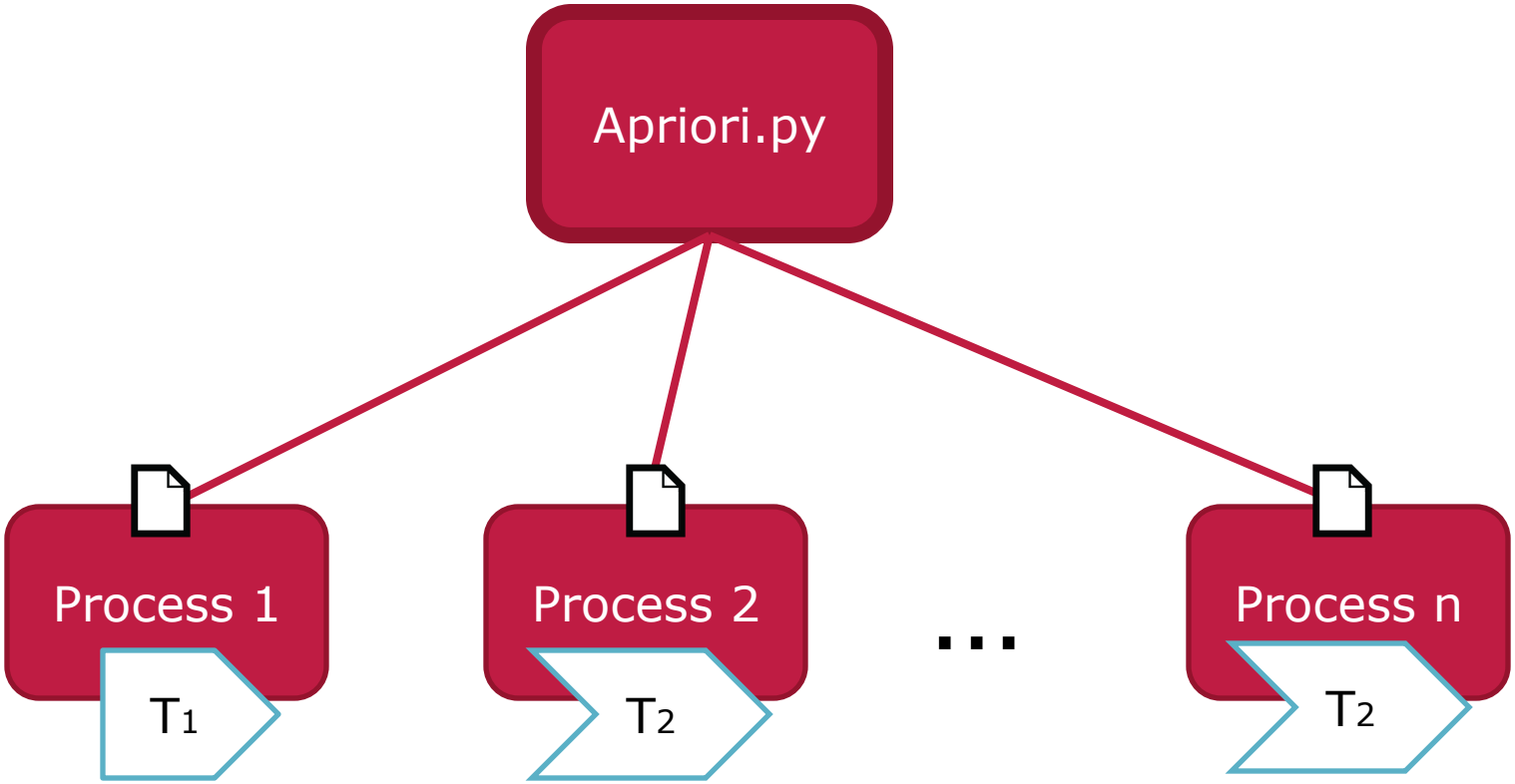
Parallelisierungsszenarien (2/3)



Parallelisierungsszenarien (2/3)



Parallelisierungsszenarien (2/3)



Analyse (2/3)

- Initialisierungsaufwand: |D|
- Aufwand in Schritt k: n * $|C_k|$
- Ergebnisgröße: n * $|C_k|$

5Mio

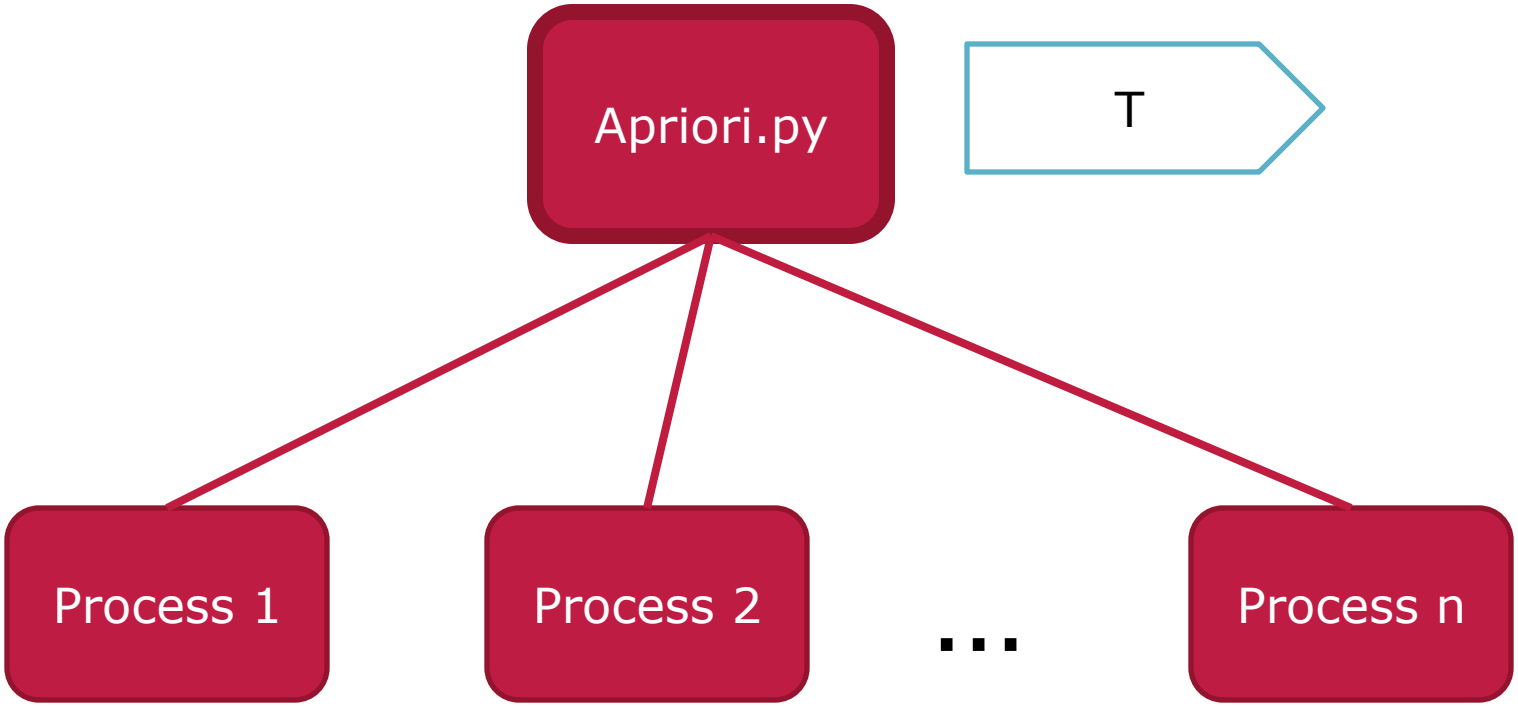
4 * 400

4 * 400

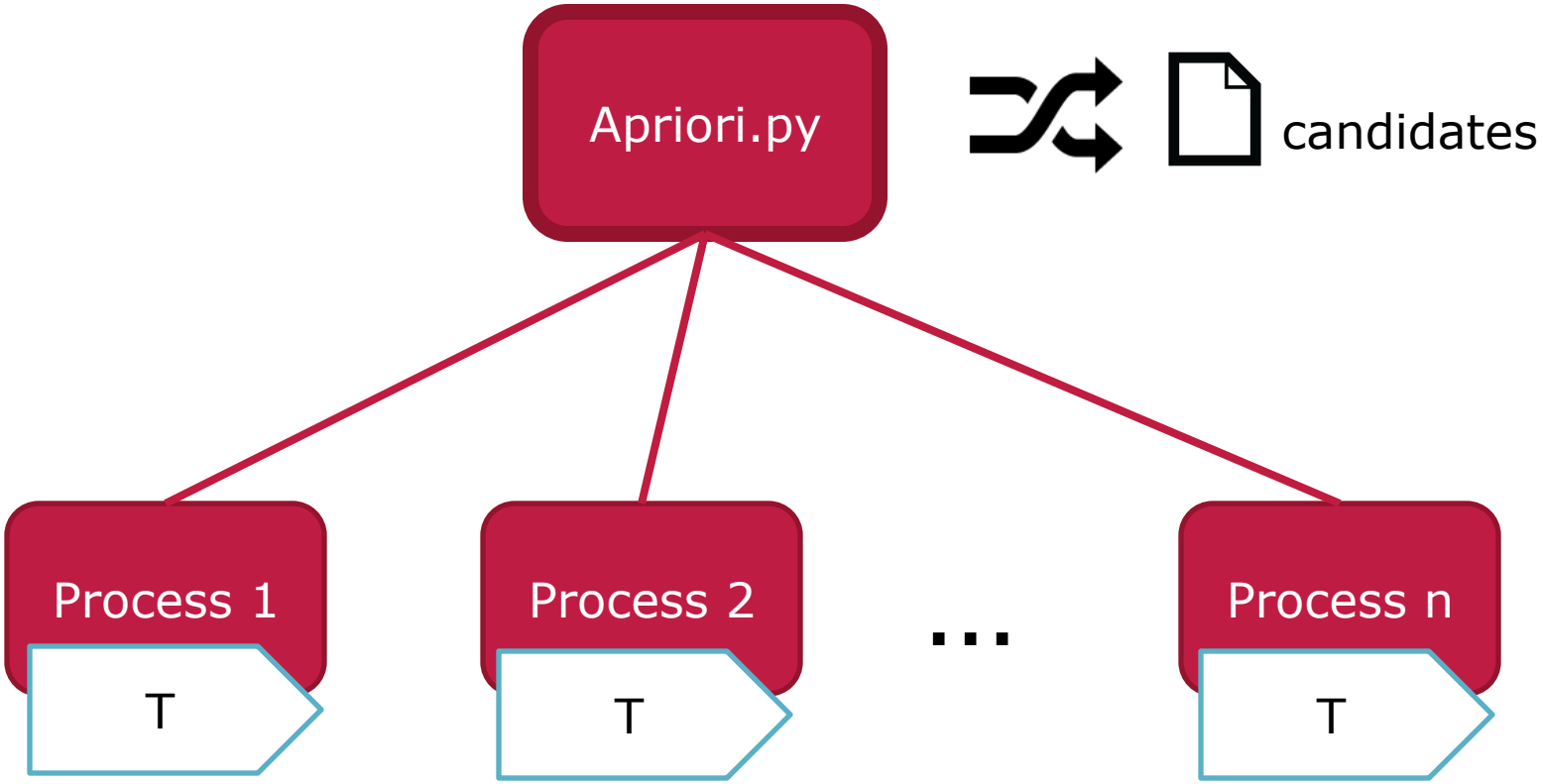
- Gesamtkosten: $|D| + \sum_k 2n|C_k|$

5Mio

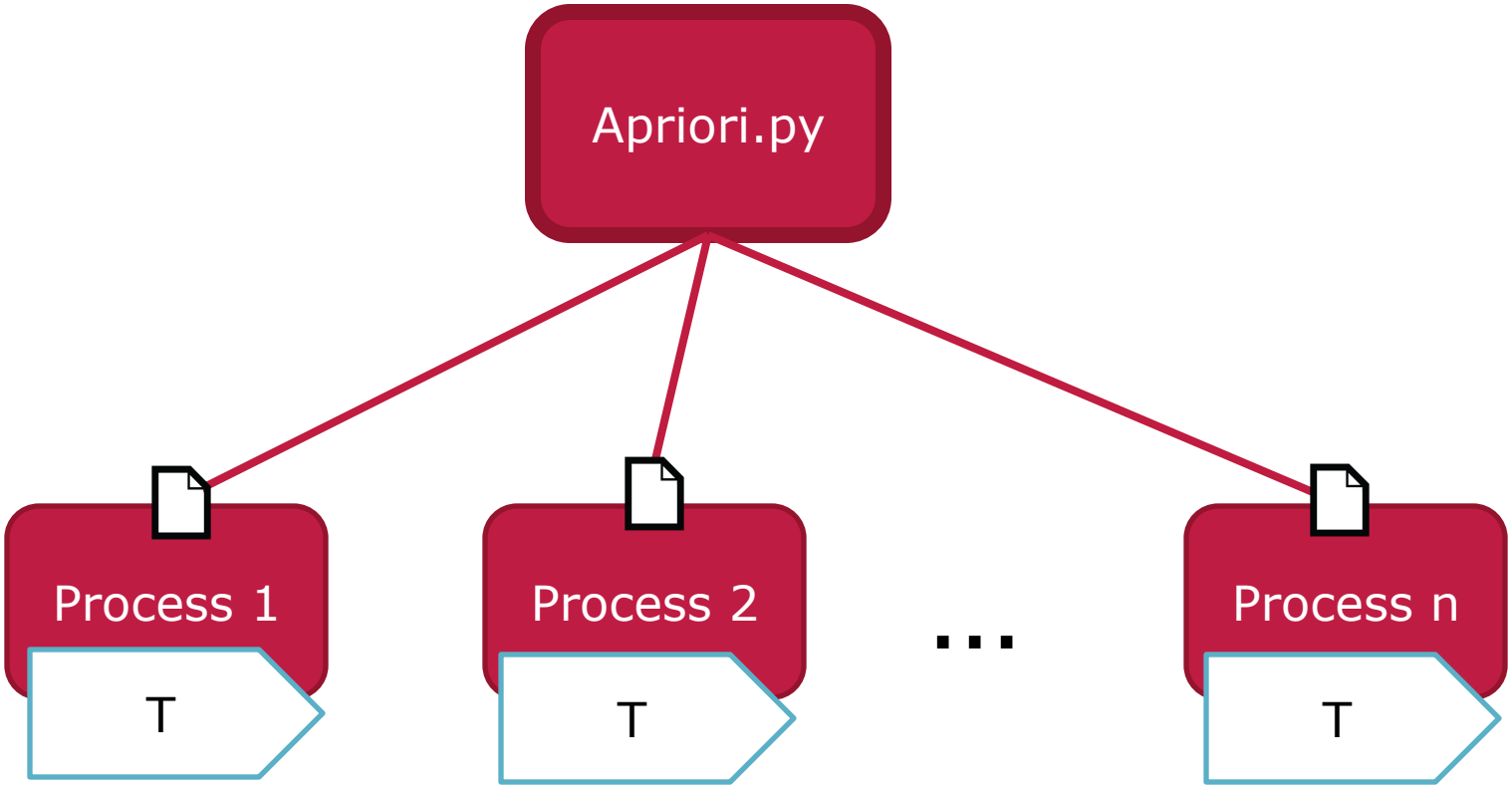
Parallelisierungsszenarien (3/3)



Parallelisierungsszenarien (3/3)



Parallelisierungsszenarien (3/3)



Analyse (3/3)

- Initialisierungsaufwand: n * |D|
- Aufwand in Schritt k: $|C_k|$
- Ergebnisgröße: $|C_k|$

- Gesamtkosten: $n|D| + \sum_k 2|C_k|$

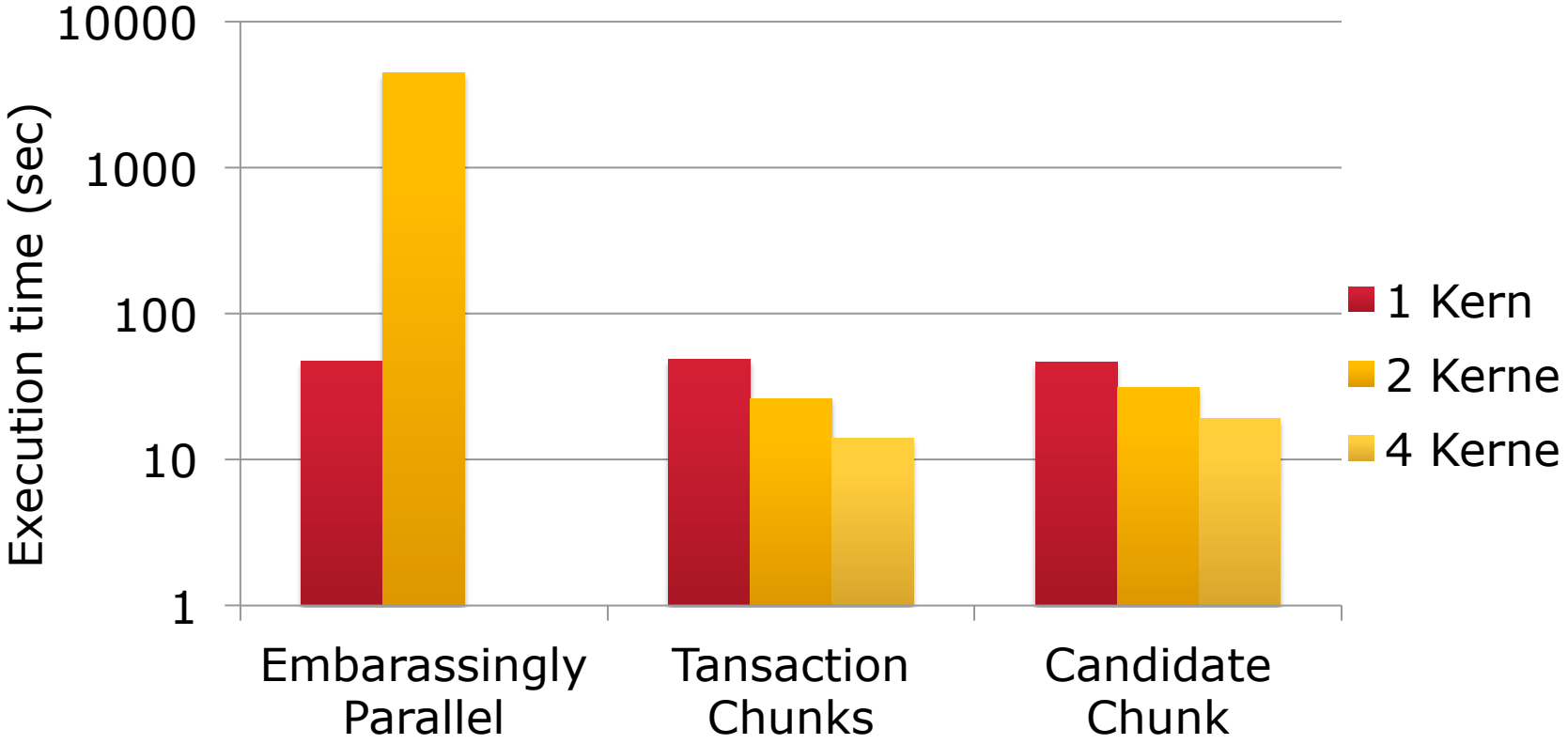
4 * 5Mio

400

400

20Mio

Theorie vs. Praxis



$$\sum_k (|D|+1) * |C_k|$$

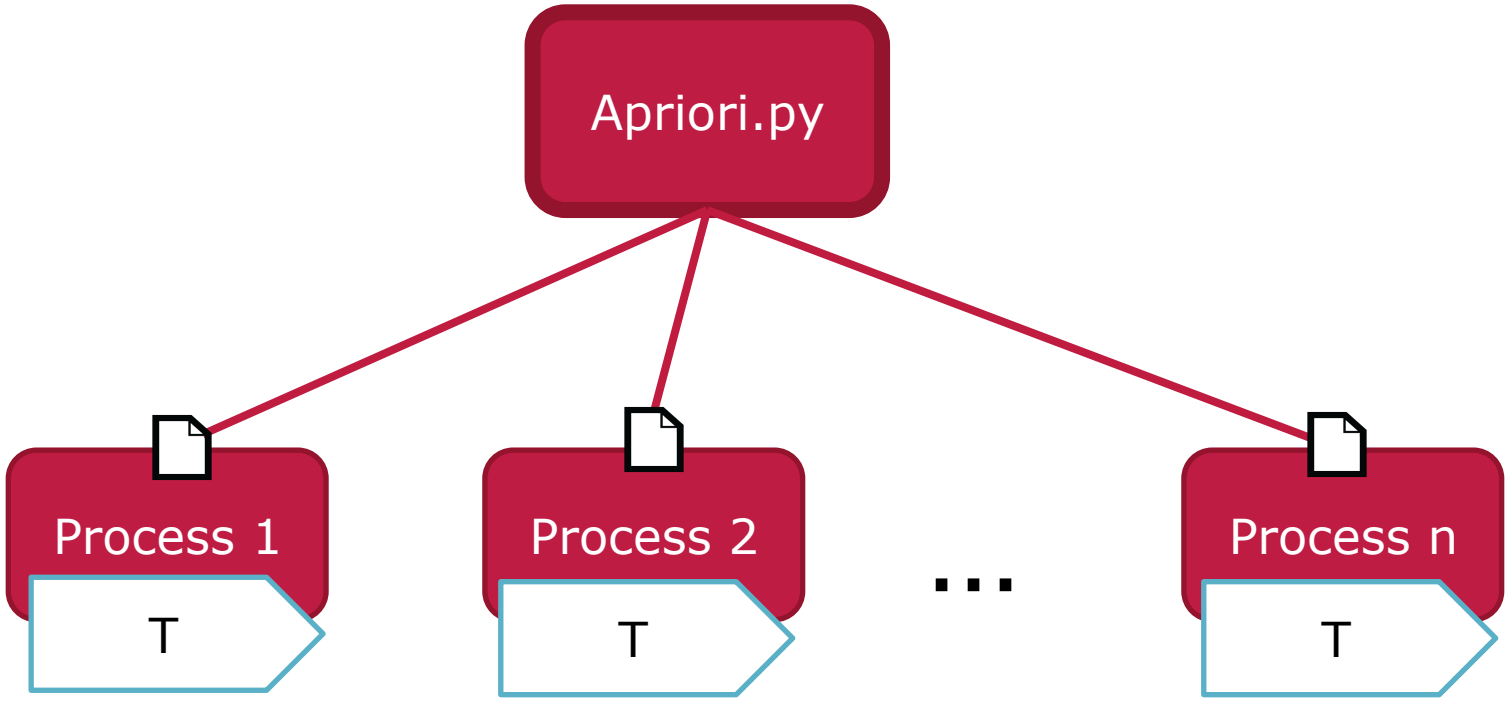
$$|D| + \sum_k 2n|C_k|$$

$$n|D| + \sum_k 2|C_k|$$

Implementierung (1)

- Map/Reduce
  - Map(candidates, transactions) → list(candidate,count)
  - Reduce(candidate, list (counts)) → list(candidates)

Implementierung (2)



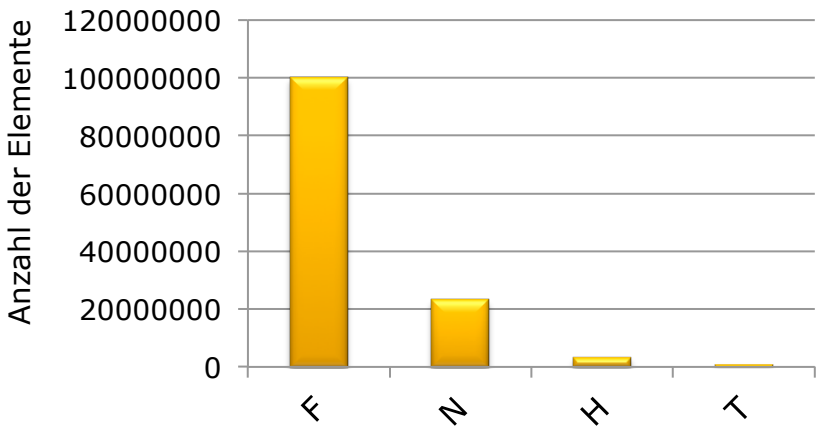
- Update der Transaktionen
  - Würde Rückübertragung aller Transaktionen benötigen
  - Daher nicht mehr AprioriTID sondern Apriori

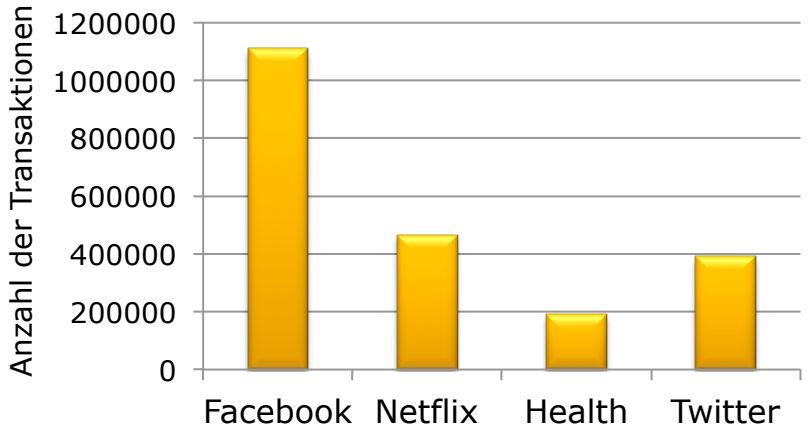
Fazit & Ideas

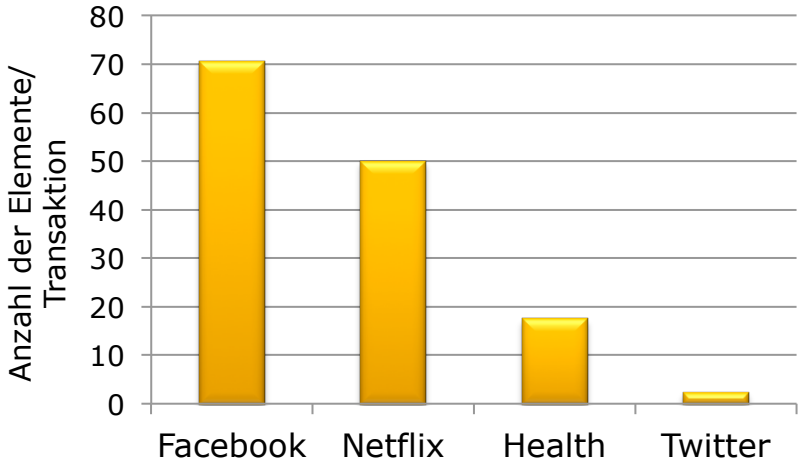
- Parallelisierung in anderen Programmiersprachen könnte mit Threads implementiert werden
  - Weniger Kommunikationsoverhead
  - Geringere Context Switch Kosten
  - Aber ggf. Wartezeiten wegen locking
- Parallelisierung auf Hadoop
- Hoher Data Parallelism

  ➜ CUDA/OpenCL

Datensets

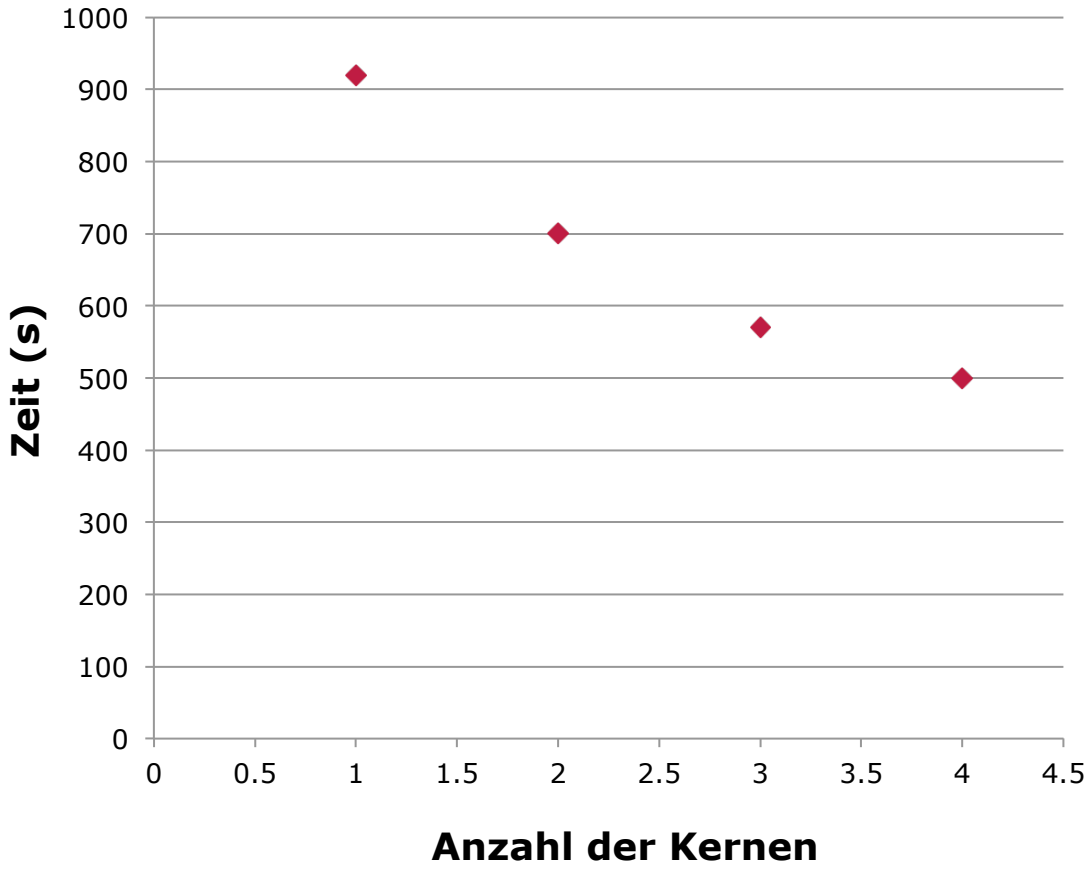

Anzahl der Elemente
120000000
100000000
80000000
60000000
40000000
20000000
0
F
N
H
T


Anzahl der Transaktionen
1200000
1000000
800000
600000
400000
200000
0
Facebook
Netflix
Health
Twitter


Anzahl der Elemente/ Transaktion
80
70
60
50
40
30
20
10
0
Facebook
Netflix
Health
Twitter

Benchmark (Netflix)

- |D| = 200.000
- AVG(|T|) = 52,6
- N = 10.232.001
- 40% speedup

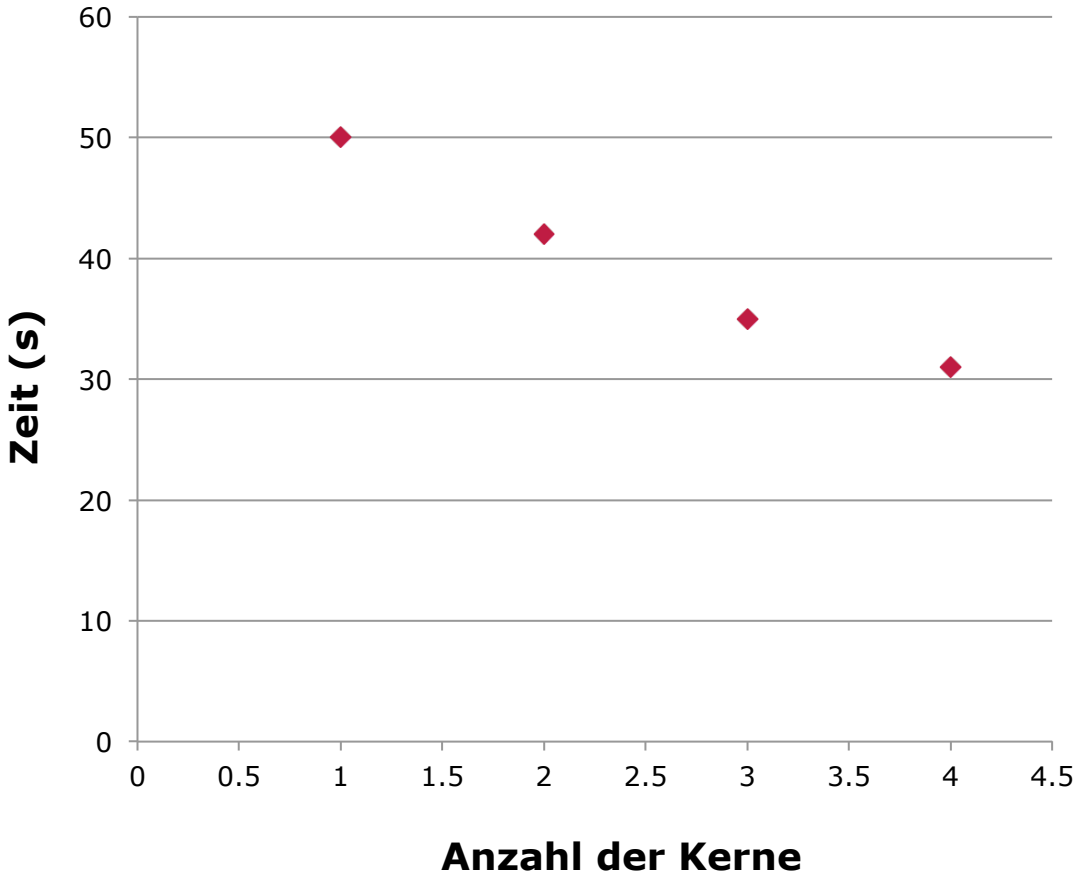


Hardware:

- Core i7-2600 Quad-Core
- 16GB RAM

Benchmark (Twitter)

- |D| = 200.000
- AVG(|T|) = 2,4
- N = 480.000
- 40% speedup

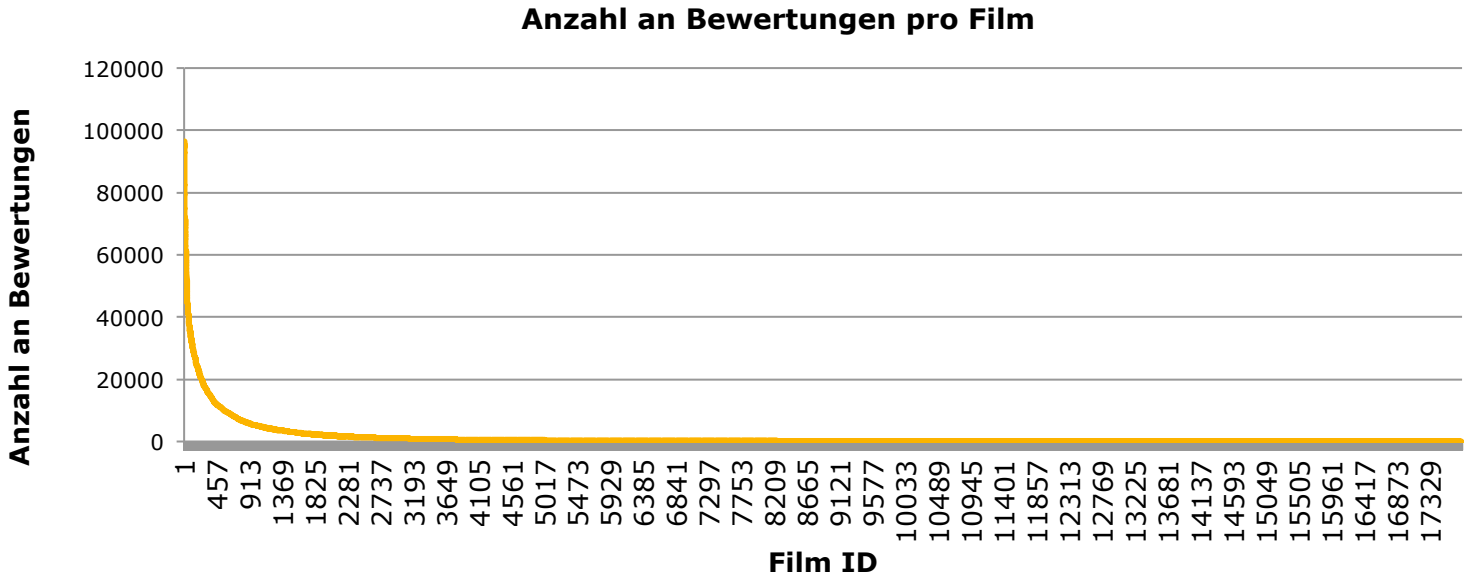


Hardware:

- Core i7-2600 Quad-Core
- 16GB RAM

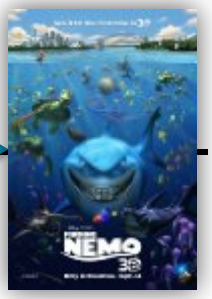
Auffinden interessanter Regeln

Pirates of the Caribbean -> LoR II



Lift

$$lift(X \rightarrow Y) = \frac{\sup(X \cup Y)}{\sup(X) * \sup(Y)}$$

$$lift(\text{  } \rightarrow \text{  }) = \frac{5.78\%}{7.2\% * 18\%} = 4.46$$

Errechnete Werte: 3,7 – 7,5

Conviction

$$conv(X \rightarrow Y) = \frac{1 - \sup(Y)}{1 - conf(X \rightarrow Y)}$$

$$conv(\text{  }) = \frac{1 - 0.18}{1 - 0.77} = 3.55$$

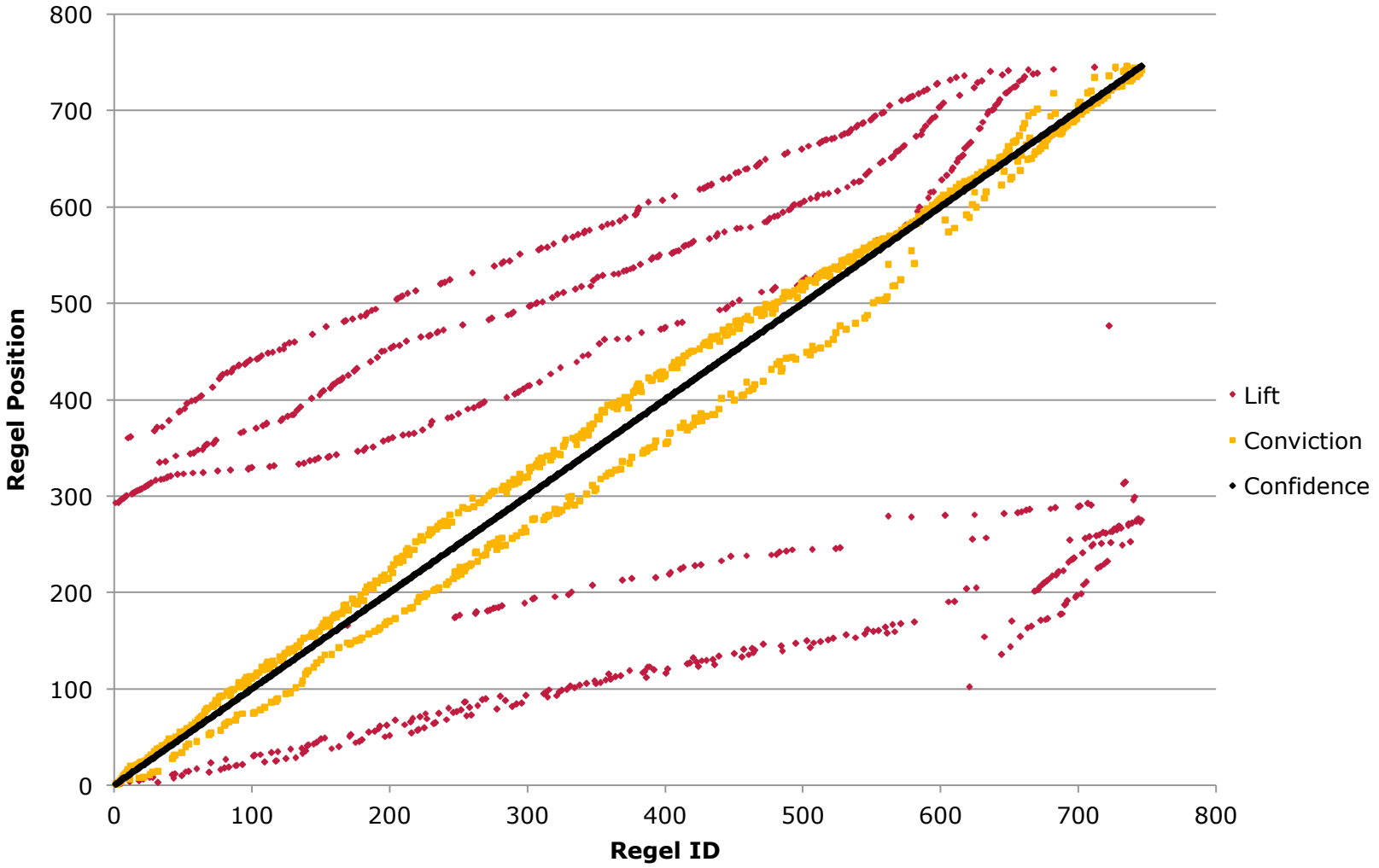
Errechnete Werte: 3,5 - 25.32

Ergebnisse

Support: 5,3 %
Confidence: 96,5 %
Lift: 7,5
Conviction: 25,3

Support: 5%
Confidence: 75%
746 Regeln

Einfluss von Lift und Conviction



Vergleich zu IMDb

Stichprobe (N=30):

21/30 = 70% stimmen mit IMDb überein

- Nachfolger
- Gleiches Genre

Quellen

- Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market basket data.
- P. Becuzzi, M. Coppola and M. Vanneschi. Mining of Association Rules in Very Large Databases: A Structured Parallel Approach*
- Ali Tarhini: Parallel Apriori algorithm for frequent pattern mining. http://alitarhini.wordpress.com/2011/02/26/parallel-apriori-algorithm-for-frequent-pattern-mining/ [Stand 3. Juli 2012]
- Anuradha.T, Satya Pasad R, S.N.Tirumalarao. Parallelizing Apriori on Dual Core using OpenMP. International Journal of Computer Applications, Volume 43
- R. Agrawal, R. Srikant: "Fast Algorithms for Mining Association Rules", Proc. of the 20th Int'l Conference on Very Large Databases, Santiago, Chile, Sept. 1994