
UNIVERSITY OF BAMBERG
OTTO-FRIEDRICH-UNIVERSITÄT BAMBERG
S.OTTO
PAT.UNIV.BAMBERG

Silentium!

DT-DB4MLKD-B
Silentium!

1) Einführung

2) Was ist „Noise“?

3) Database Engine System Konfiguration
    3.1) Completely Fair Scheduling
    3.2) Separated Schedulung

4) Experiments: Simulating tenant load
    4.1) Database Engine
    4.2) Data-Queries
    4.3) Simulating
    4.4) Results

5) Konsequenzen & Konklusion

Universität Bamberg

DT-DB4MLKD-B
Silentium!

1) Einführung

„Silentium! Run–Analyse–Eradicate the Noise out of the DB/OS Stack“

By Wolfgang Mauerer, Ralf Ramsauer, Edson R. Lucas F, Daniel Lohmann, Stefanie Scherzinger

Problem:
Bei der gleichzeitigen Benutzung von Datenbanken entstehen beim Abfragen dieser Latenzen.

Ziel des Papers:
Der experimentbasierte Beweis, dass es möglich ist, multiple Queries an eine Datenbank zu richten und dabei Geschwindigkeiten zu erreichen, welche vergleichbar mit Einzelabfragezeiten sind.

Universität Bamberg

DT-DB4MLKD-B
Silentium!

Operating System - Sowohl Fluch als auch Segen:
- Hardware Support
- Drivers
- System Abstraktionen

→ spezielle Betriebssysteme

→ hier jedoch der Fokus auf existierende Open-Source Komponenten & Identifikation, Analyse und Behandlung der Gründe für Noise

Universität Bamberg

DT-DB4MLKD-B
Silentium!

2) Was ist Noise?

<u>Noise in der Datenkommunikation:</u>

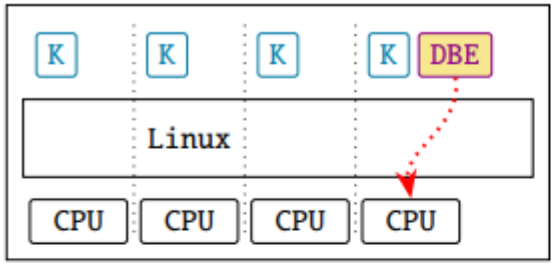
“In der Netzwerkkommunikation bezeichnet Noise unerwünschte Störungen, die die Übertragung von Daten beeinträchtigen können.
Dies kann beispielsweise durch elektromagnetische Interferenzen, fehlerhafte Kabelverbindungen oder schlechte Signalqualität verursacht werden.
Wenn Noise in einer Netzwerkkommunikation auftritt, kann dies zu Datenverlust, Übertragungsfehlern oder einer Beeinträchtigung der Leistung führen.“

Noise-Quellen:
1) Andere Prozesse und System Services welche die CPU benutzen
2) unkontrollierte CPU Perfomanzoptimierungen (Caches, Pipelines, etc)
3) Ressourcenteilung (zB memory bus)

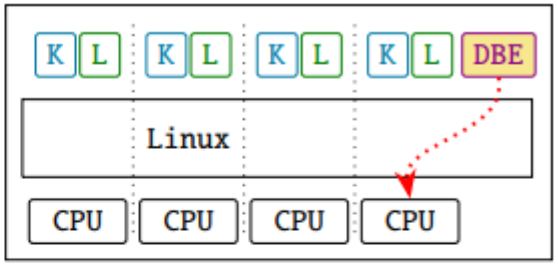
→ Durch die Noise wird der Determinismus defizitär

DT-DB4MLKD-B
Silentium!

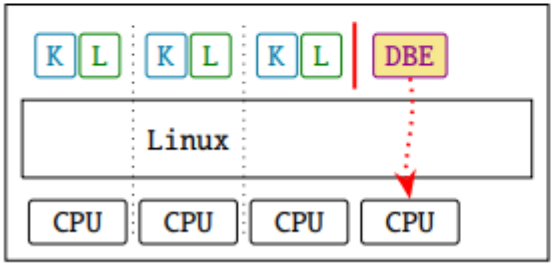
3) Database Engine (DBE) System Konfiguration



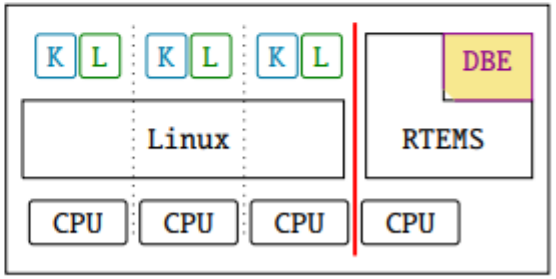
(a) No load.



(b) Load.



(c) Shielding the database engine.



(d) Isolation via hardware partitioning.

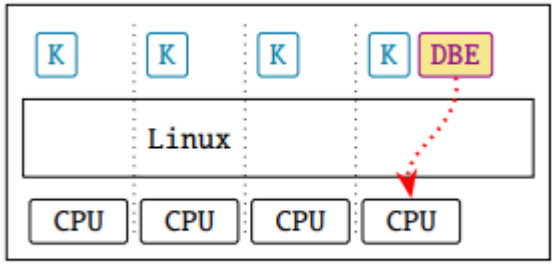
DT-DB4MLKD-B
Silentium!

CFS - Linux Completely Fair Scheduler
- virtual runtime in Nanosekunden
- Auswahl des Prozesses mit niedrigster CPU-Laufzeit
- Beispiel:
  - Tenant 1 hat einen Prozess
  - Tenant 2 hat zwei Prozesse
  - → CFS teilt jeweils 50% der CPU an 1 & 2

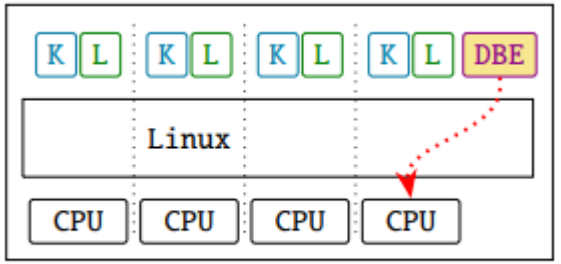
No Load
- DBE läuft alleine
- einer dedizierten CPU zugewiesen

Load
- Maximle Auslastung
- Alle Tenants gleichbehandelt

Load/FIFO
- Maximle Auslastung
- First In First Out
- Durch incoming interrupts noch immer störbar



(a) No load.



(b) Load.

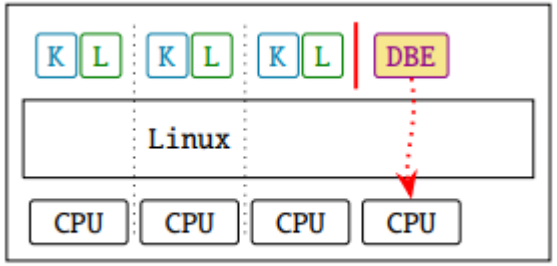
DT-DB4MLKD-B
Silentium!

Shielding
- Alle existierenden Tasks & Threads einer CPU zugewiesen
- Verhindert Benutzung der dedizierten CPU
- Zusätzlich: Interupts nur an andere CPUs

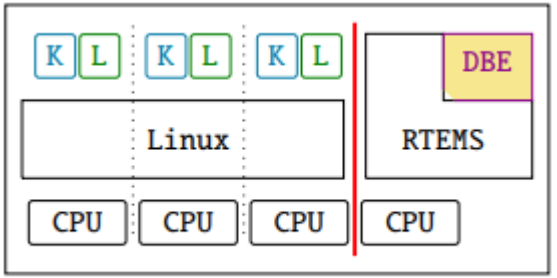
Partitioning (Jailhouse hypervisor)
- Trennung auf Systemarchitekturebene
- CPUs, Speicher, Geräte, Caches, System Buses
- Gering höherer Overhead
- Anwendung in sicherheitskritischen Szenarios

„Bare Metal“
- x86 & ARM Architektur
- Einsatz bei „Long Tail Latency“ Problemen
- Portable DBToaster läuft mit minimal-OS RTEMS (real-time executive for multiprocessor systems) mit C++

→ Erlaubt den Vergleich der Variationen



(c) Shielding the database engine.



(d) Isolation via hardware partitioning.

DT-DB4MLKD-B
Silentium!

4) Experiments: Simulating tenant load

- 4.1) Database Engine
- 4.2) Data-Queries
- 4.3) Simulating
- 4.4) Results

Universität Bamberg

DT-DB4MLKD-B
Silentium!

4.1) Database Engine

DBToaster
- Kompiliert SQL zu C++ zur Zielplatform
- Single-Thread DBE
- Inkrementelle Updates des SQL view solange Tuple eingehen

→ SQL-to-code-compiler mit niedriger Latenz

Einsatzgebiete
- Stream Processing
- Algorithmisches Handeln
- Netzwerküberwachung

Universität Bamberg

DT-DB4MLKD-B
Silentium!

4.2) Data-Queries

Query-Auswahl
- hochvariable Latenzen
- hochvariabler rechnischer Aufwand (z.B. sub-query, multi-joins)

Finanzqueries
- Börsenhandelsaktivität
- Query „countone“ (baseline)
- Query „Axfinder AXF“ (group-by, join, selection, aggregation)
- Query „Pricespread PSP“ (join, selection, aggregation)

TPC-H-Queries
- Benchmark des Transaction Processing Performance Council
- Verschiedene selections, aggregations sowie ein join

DT-DB4MLKD-B
Silentium!

C1

```
SELECT count (1) FROM bids;
```

axfinder

```
SELECT b.broker_id,
 SUM(a.volume+(-1*b.volume)) AS axfinder
FROM bids b, asks a
WHERE b.broker_id = a.broker_id
  AND ((a.price+((-1) * b.price)>1000)
  OR (b.price+((-1) * a.price)>1000))
GROUP BY b.broker_id;
```

TPCH Q11a

```
SELECT ps.partkey,
  SUM(ps.supplycost * ps.availqty)
  AS query11a
FROM  partsupp ps, supplier s
WHERE ps.suppkey = s.suppkey
GROUP BY ps.partkey;
```

pricespread

```
SELECT SUM(a.price + (-1*b.price))
       AS psp
FROM bids b, asks a
WHERE (b.volume > 0.0001 *
 (SELECT SUM(b1.volume) FROM bids b1))
  AND (a.volume > 0.0001 *
 (SELECT SUM(a1.volume) FROM asks a1));
```

TPCH Q6

```
SELECT SUM(l.extendedprice*l.discount)
       AS revenue
FROM lineitem l
WHERE l.shipdate>=DATE('1994-01-01')
  AND l.shipdate<DATE('1995-01-01')
  AND ( l.discount BETWEEN (0.06-0.01)
        AND (0.06+0.01) )
  AND l.quantity<24;
```

TPCH Q1

```
SELECT returnflag, linestatus,
  SUM(quantity) AS sum_qty, SUM(extendedprice) AS sum_base_price,
  SUM(extendedprice*(1-discount)) AS sum_disc_price,
  SUM(extendedprice*(1-discount)*(1+tax)) AS sum_charge,
  AVG(quantity) AS avg_qty, AVG(extendedprice) AS avg_price,
  AVG(discount) AS avg_disc, COUNT(*) AS count_order
FROM lineitem
WHERE shipdate<=DATE('1997-09-01')
GROUP BY returnflag, linestatus;
```

Universität Bamberg

DT-DB4MLKD-B
Silentium!

4.3) Simulating

DBToaster Log Zeitstempel=

$$\frac{\text{Latenz per N-Input Tuples}}{\text{N-Tuples}}$$

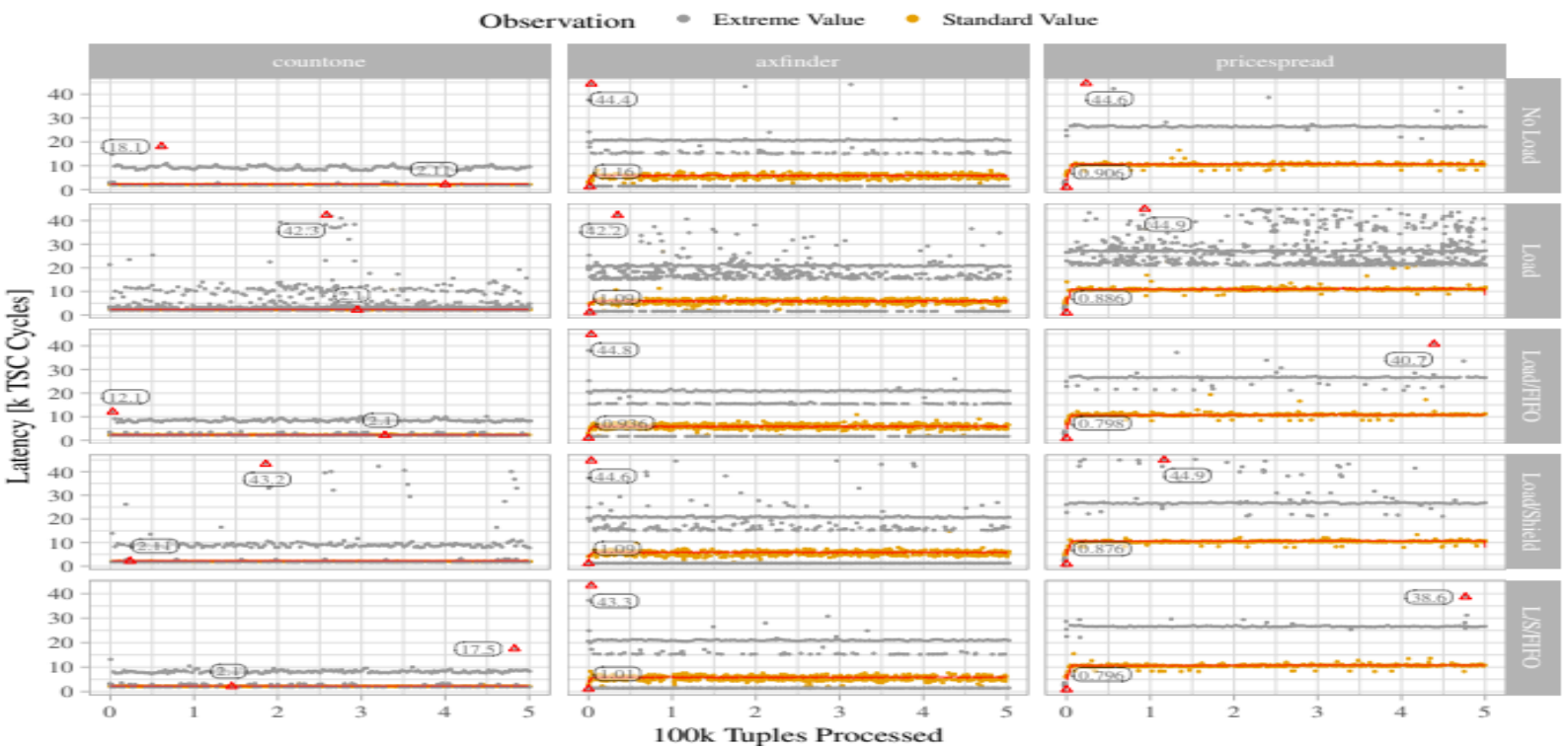
Timestamps
- POSIX API
  (clock_gettime with CLOCK_MONOTONIC)
- x86 Time Stamp Counter
  (TSC via DBToaster)

Tenant load
- 6 synthetische Workloads (stress-ng)

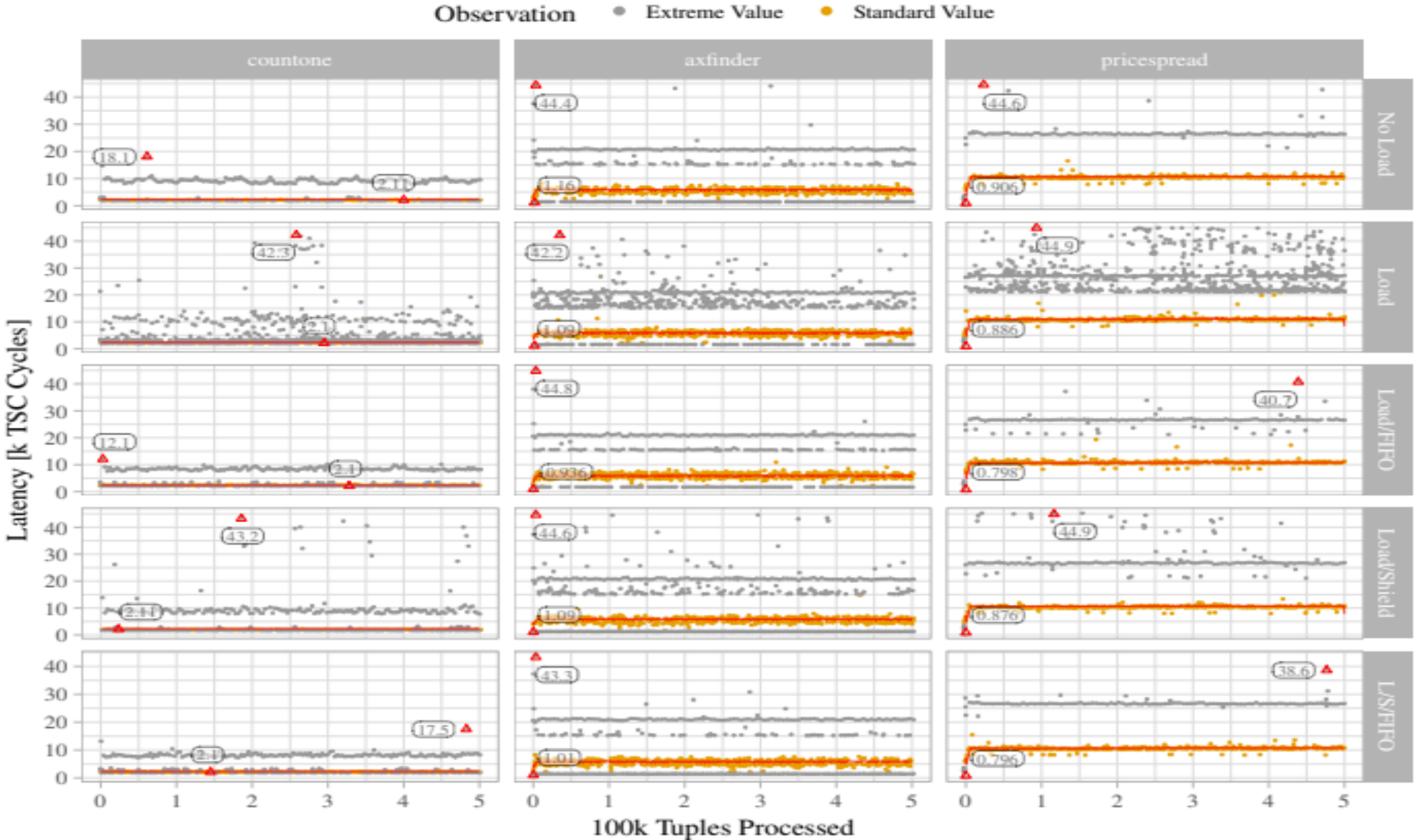
Universität Bamberg

DT-DB4MLKD-B
Silentium!

4.4.1) Results - Noise and Determinism: Finance Queries

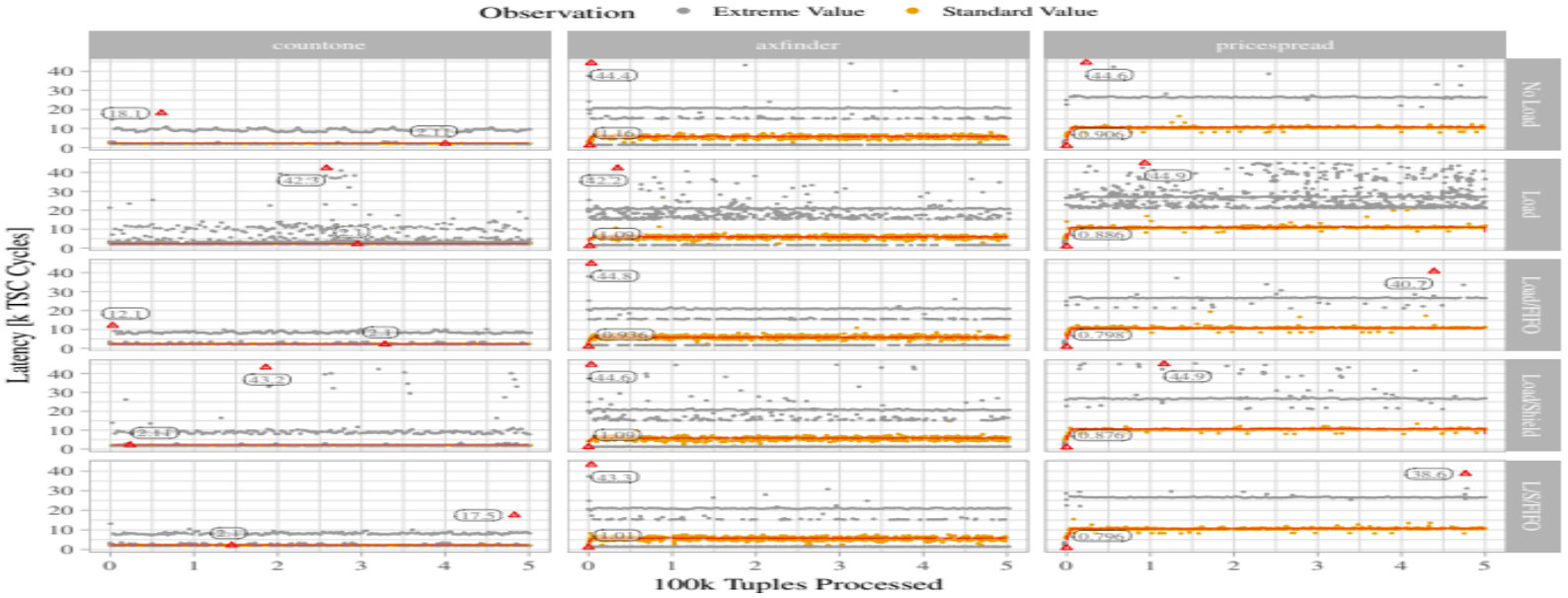


Universität Bamberg


Observation
Extreme Value
Standard Value
countone
axfinder
pricespread
No Load
Load
Load/FIFO
Load/Shield
LS/FIFO
Latency [k TSC Cycles]
100k Tuples Processed
18.1
2.11
44.4
1.16
44.6
0.906
42.3
2.1
42.2
1.09
44.9
0.886
12.1
2.4
44.8
0.936
40.7
0.798
43.2
2.44
44.6
1.09
44.9
0.876
17.5
2.4
43.3
1.01
38.6
0.796

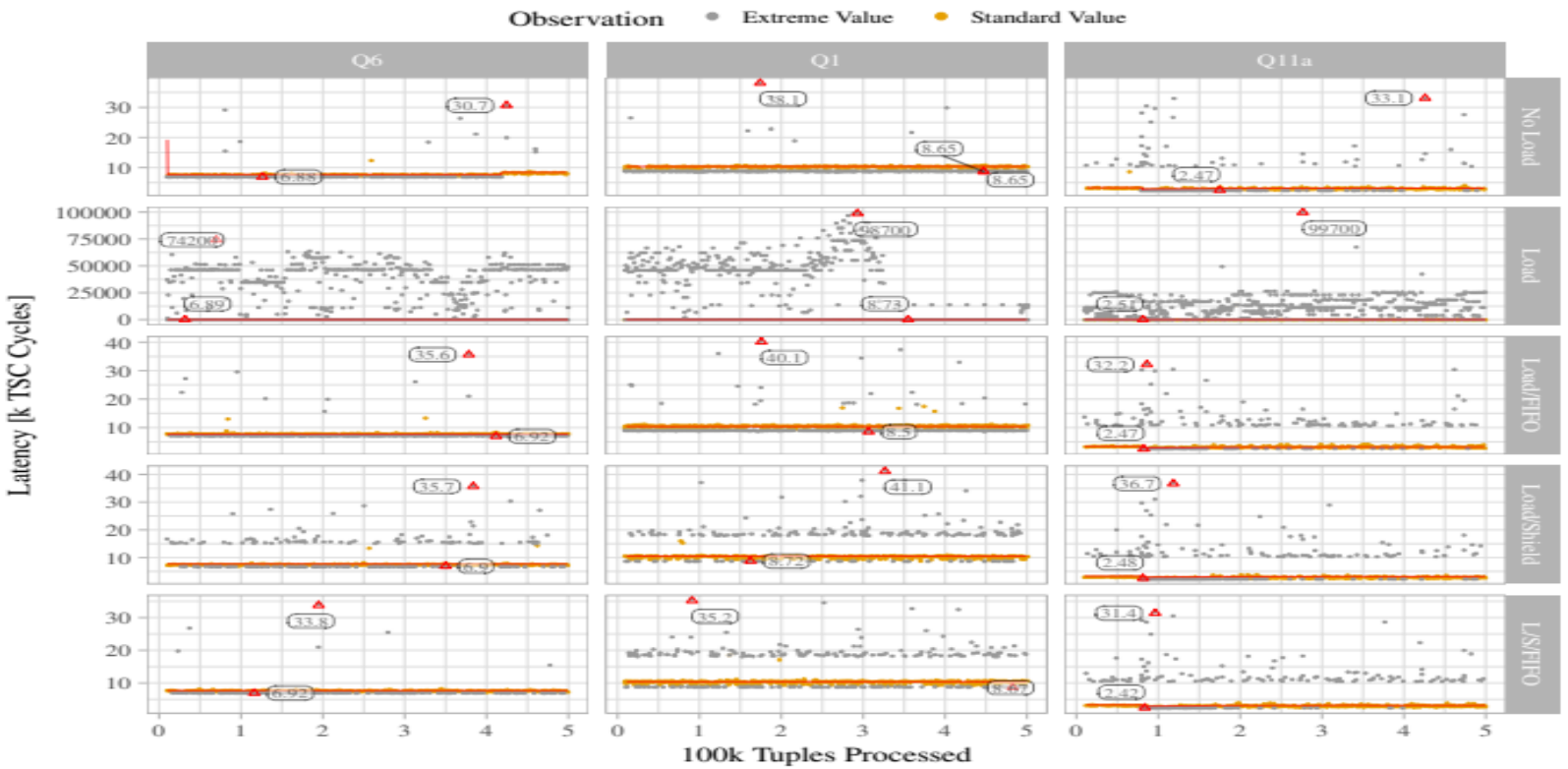
DT-DB4MLKD-B
Silentium!

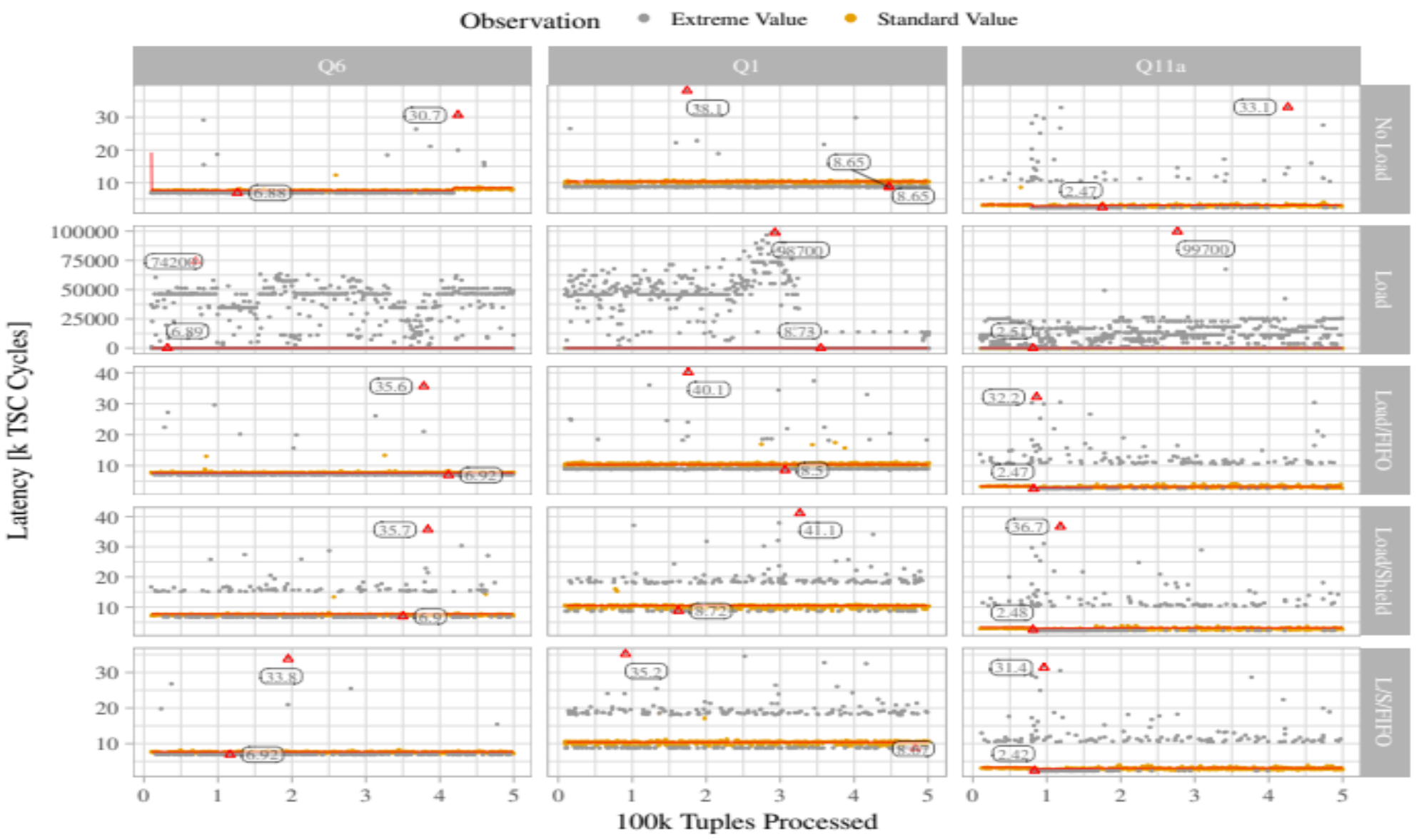
Statistische Auswertung
- Ausreißer bis 4fache Standardabweichung
- Keine direkte Beziehung zwischen Query-Komplexität und Noise
- Aber eine Beziehung zwischen Query-Komplexität und durchschnittlicher Performance
- Noise in abnehmender Reihenfolge: Load/Shield, Load/FIFO, Load/Shield/FIFO



DT-DB4MLKD-B
Silentium!

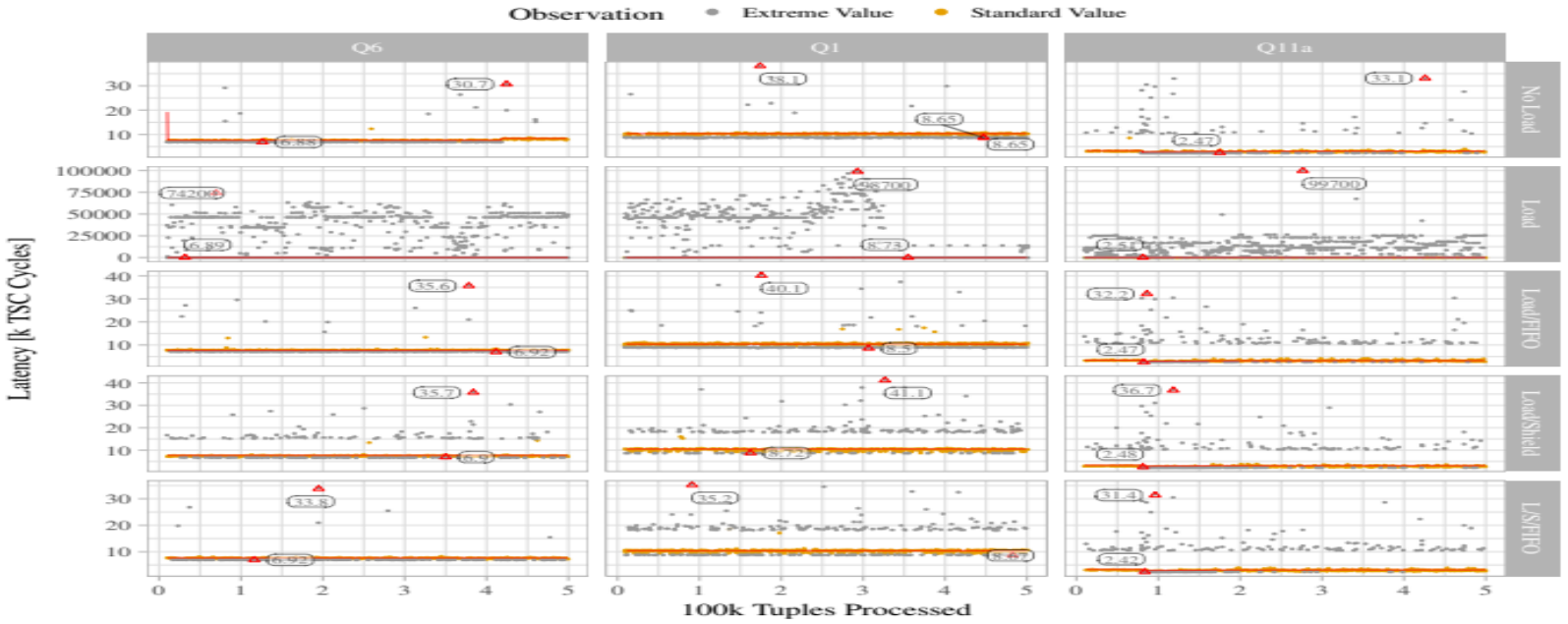
4.4.2) Results - Noise and Determinism: TCP-H Queries




Observation
Extreme Value
Standard Value
Q6
Q1
Q11a
No Load
Load
Load/FIFO
Load/Shield
L/S/FIFO
Latency [k TSC Cycles]
100k Tuples Processed
30.7
6.88
38.1
8.65
8.65
33.1
2.47
74200
6.89
98700
8.73
99700
2.51
35.6
6.92
40.1
8.5
32.2
2.47
35.7
6.9
41.1
8.72
36.7
2.48
33.8
6.92
35.2
8.47
31.4
2.42

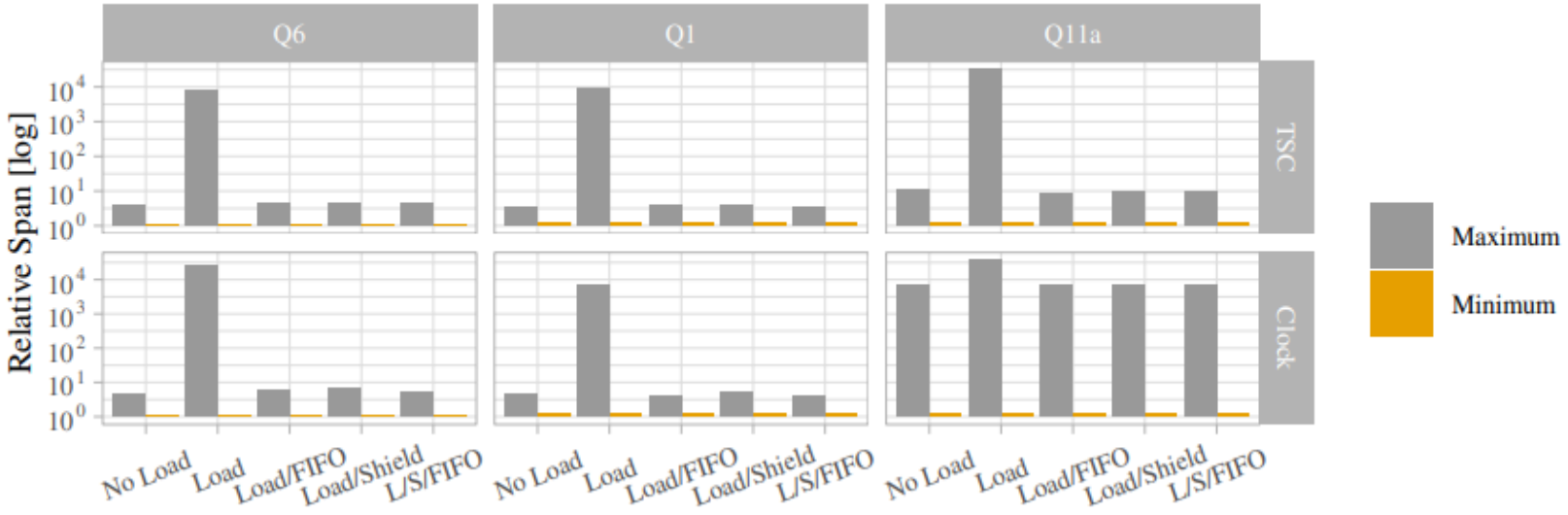
DT-DB4MLKD-B
Silentium!

Statistische Auswertung
- Generelle Beobachtungen Identisch
- Ausreißer bis 10^3fache Standardabweichung
- Sehr hohe Varianz
- Relative Zeitspanne sehr hoch



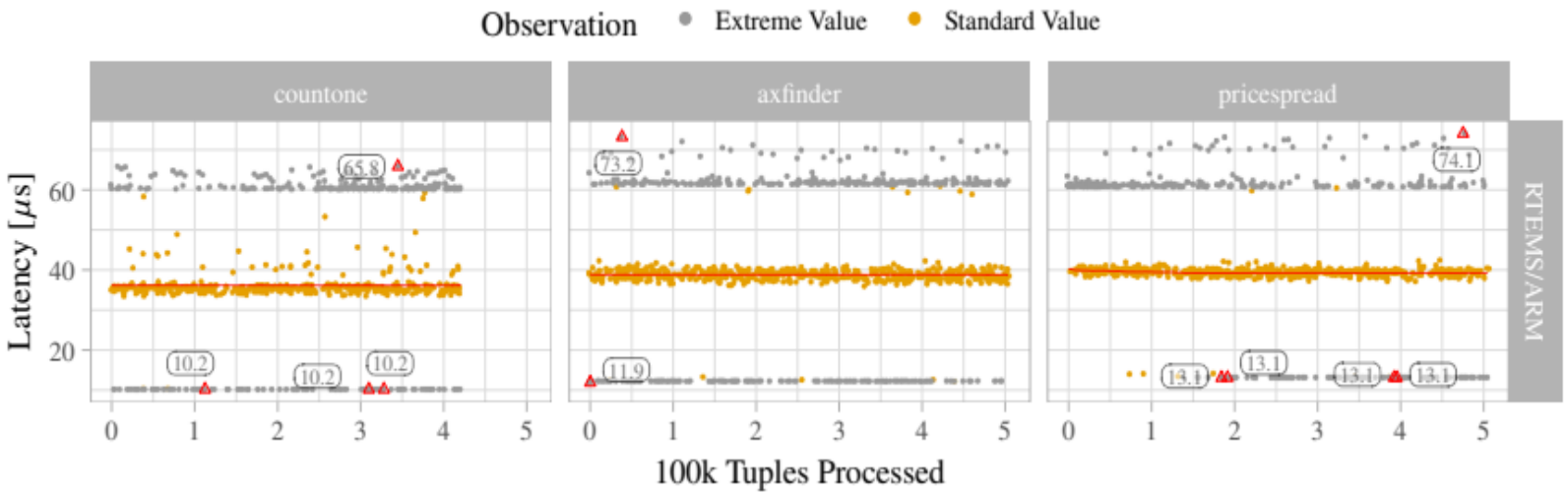
DT-DB4MLKD-B
Silentium!

4.4.2) Results - Noise and Determinism: TCP-H Queries
- Verteilung in Load Szenario



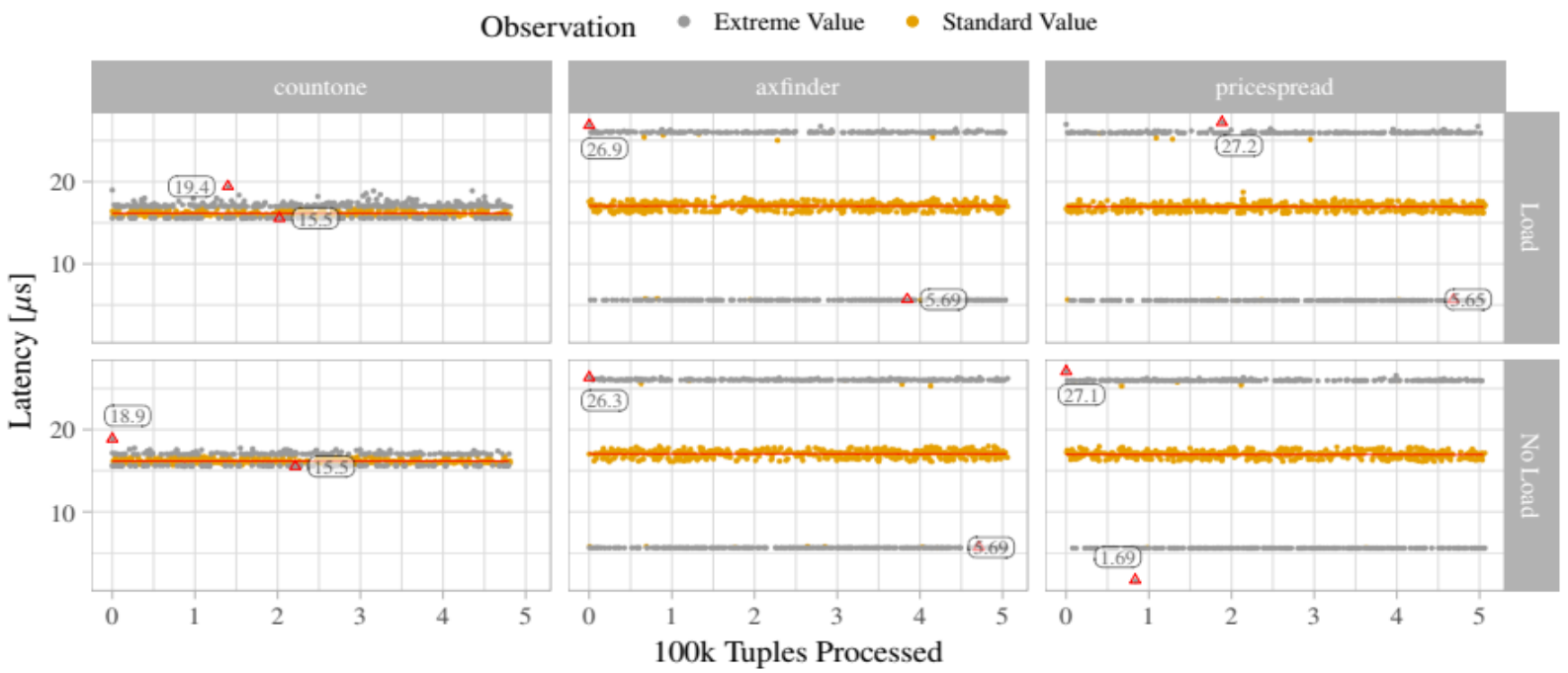
DT-DB4MLKD-B
Silentium!

4.4.3) Results - CPU-Noise



DT-DB4MLKD-B
Silentium!

4.4.3) Results - CPU-Noise



DT-DB4MLKD-B
Silentium!

5) Konsequenzen & Konklusion

Hauptursache Noise
→ Noise- und Zeitmessung selbst

→ Keine Notwendigkeit OS/Hardware speziell zu kreieren, sondern weitere Forschung über die Gründe für Noise nötig
→ Existierende OS-Solutions wie zb Jailhouse Partitioning sollten mehr in DB-Systeme integriert werden
→ Silentium!-Paper addressiert CPU Noise und Ignoriert I/O Noise

Universität Bamberg

DT-DB4MLKD-B
Silentium!

Quellen:

- Silentium! Run–Analyse–Eradicate the Noiseout of the DB/OS Stack
Mauerer et al
https://dl.gi.de/server/api/core/bitstreams/1300ff5c-8db1-4770-85e9-27c7ed128ca1/content

- Noise and the Common Sense Informatic Situation for a Mobile Robot
Murray Shanahan (1996)
https://www.doc.ic.ac.uk/~mpsha/roboticsAAAI96.pdf

- Noise, Non-Determinism and Spatial Uncertainty
Murray Shanahan (1997)
https://www.doc.ic.ac.uk/~mpsha/roboticsAAAI97.pdf

- https://www.tpc.org/tpch/

„Silentium!“ ex!

Fragen incipit!