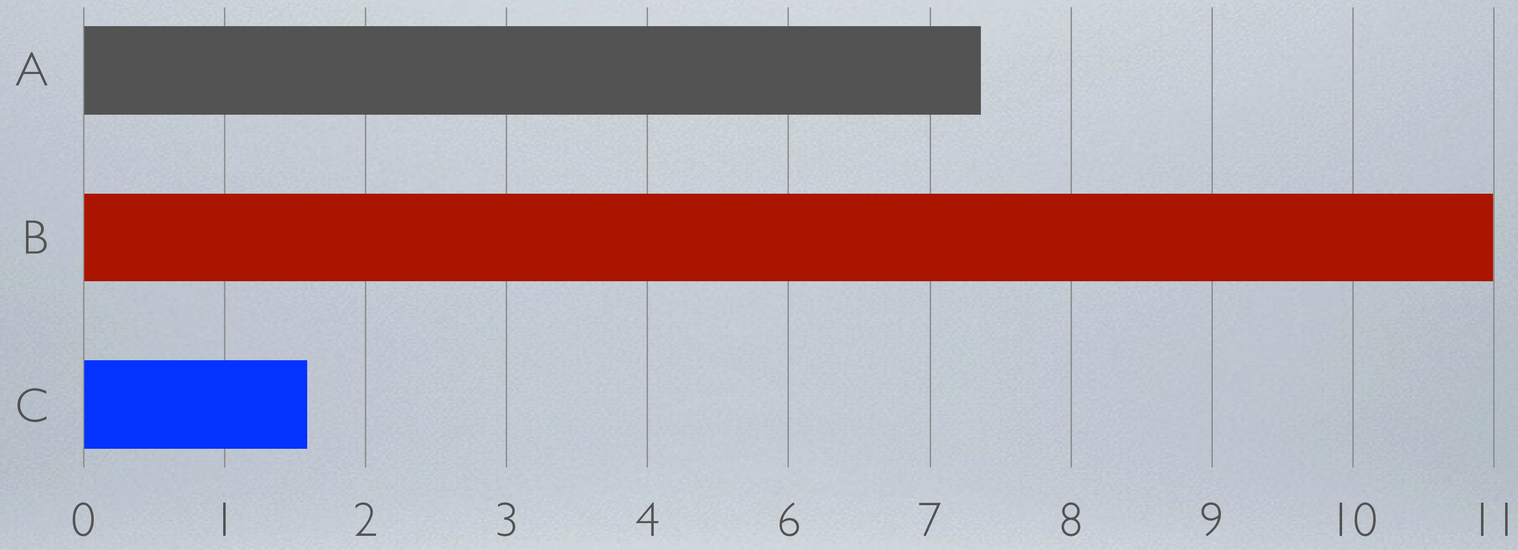
Schulz IT Services
Gesellschaft für Datenbank- und Softwareentwicklung mbH



DOAG Konferenz 2014

1000 mal schneller

Praxisgeschichten aus der Oracle-Datenbankwelt

Matthias Schulz

Selbständiger Software- und Datenbankentwickler:

- Consulting
- Schulungen
- Workshops
- Blog: oracledeli.wordpress.com

Geschäftsführer:

Schulz IT Services GmbH

Tauberstraße 28
90449 Nürnberg
Telefon (0911) 3849090

www.schulz-it-services.de
schulz@schulz-it-services.de

Matthias Schulz

Schwerpunkt Oracle

Entwicklung und Tuning:

- Datenbankdesign
- OLTP, OLAP, ETL
- SQL
- PL/SQL
- APEX
- Java in der Datenbank

Spezialthemen:

- Programmoptimierung (Datenbanknutzung)
- Exadata Database Machine
- Exasol Exasolution
- Advanced Queueing
- Replikation
- objektorientiertes PL/SQL

Schulz IT Services
Gesellschaft für Datenbank- und Softwareentwicklung mbH

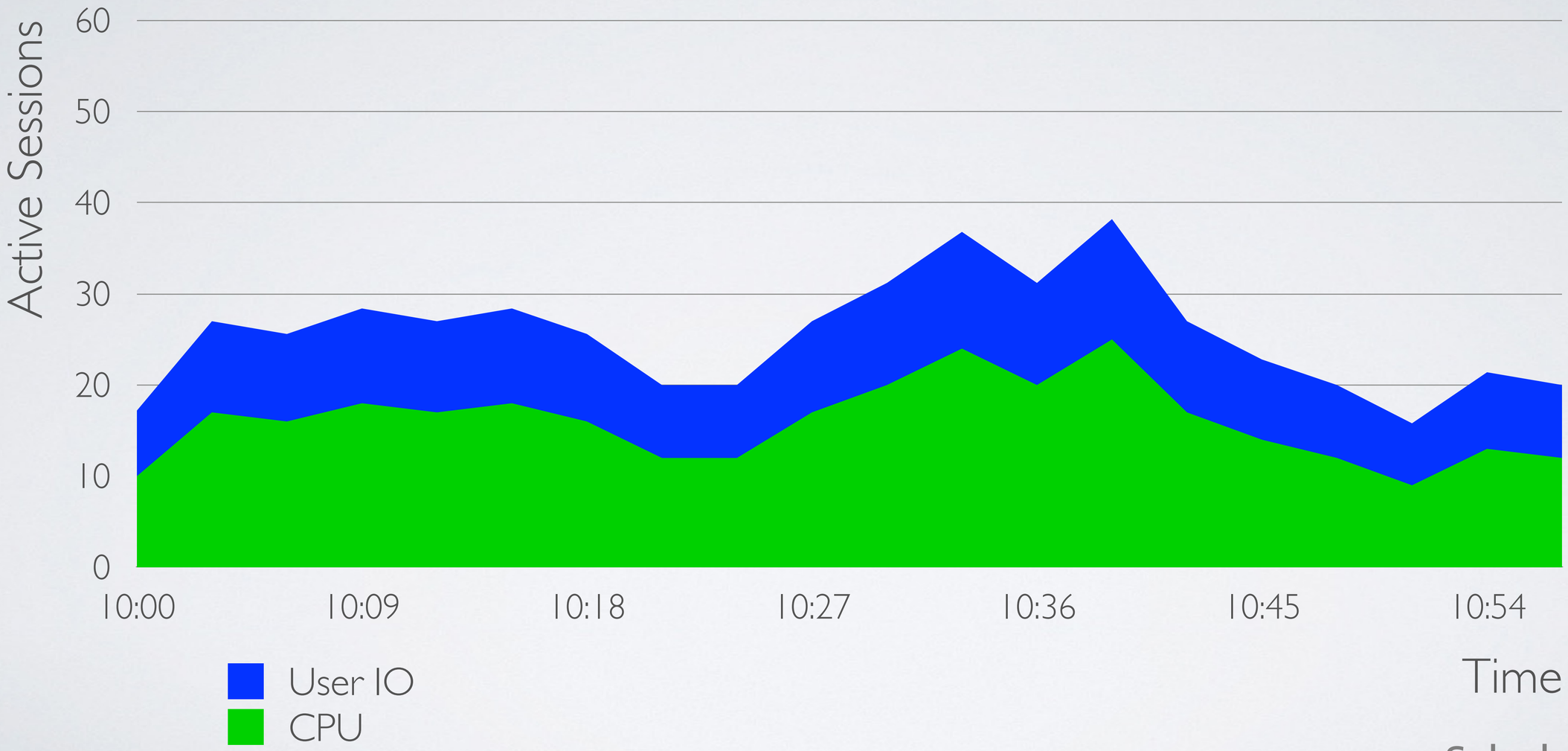
Einleitung

Echte Praxisfälle?

Alle vorgestellten Themen
basieren auf **echten Praxisfällen**,
die aus didaktischen und
datenschutzrechtlichen Gründen
abstrahiert und **vereinfacht** wurden.

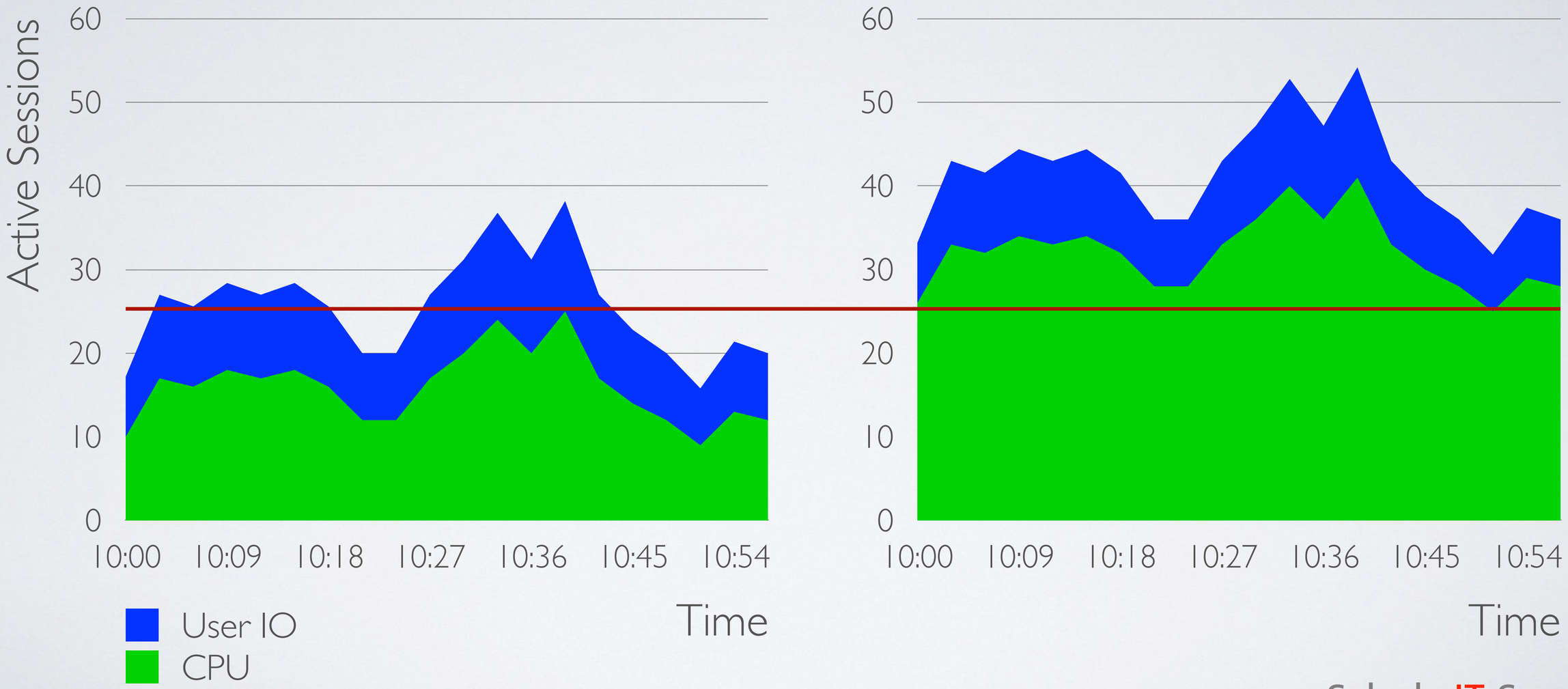
Ein fauler Apfel reicht

normale 10 Uhr Last



normale 10 Uhr Last

10 Uhr Last „heute“



Was ist los?

Was ist heute anderes als sonst?

- eigentlich nichts!

Auffällige SQL Statements?

- Top Executions? - ***nein!***
- Top I/O? - ***nein!***
- Top CPU? - ***nein!***
- … - ***nein!!!***

Nicht auffälliges in der v$sql

Row	EXECUTIONS	PARSE_ CALLS	ROWS_ PROCESSED	CPU_ SECONDS	ELAPSED_ SECONDS	APPLICATION_ WAIT_SECONDS
1	1	1	0	0,007	0,008	0
2	1	1	0	0,007	0,007	0
3	1	1	0	0,008	0,007	0
4	1	1	0	0,007	0,007	0
5	1	1	0	0,007	0,007	0
...	...	...	...	...	...	...
200.000	1	1	0	0,007	0,007	0

Besser Suchen in v$sql
mit FORCE_MATCHING_SIGNATURE

SQL ohne Bind-Variables:

```
SELECT q.FORCE_MATCHING_SIGNATURE,
       count(*) AS matches,
       sum(executions) AS executions,
       max(executions) AS max_executions,
       wm_concat(DISTINCT q.parsing_schema_name)AS parsing_schema_names,
       min(q.sql_text) AS min_text,
       max(q.sql_text) AS max_text
  FROM v$sql q
 WHERE q.FORCE_MATCHING_SIGNATURE != 0
 GROUP BY q.FORCE_MATCHING_SIGNATURE
HAVING count(*) > 10
 ORDER BY matches desc;
```

Hinweis: Bei RAC-Datenbanken **gv$sql** statt v$sql verwenden!

204.750 Executions

mit identischer FORCE_MATCHING_SIGNATURE

FORCE_MATCHING_SIGNATURE	MATCHES	EXECUTIONS	MAX_EXECUTIONS	PARSING_SCHEMA_NAMES
17877032951325766668	204.750	204.750	1	RETAIL_PUBLISH

MIN_TEXT
SELECT /* xxx */ * FROM items_test$2 WHERE INBOX_ID = 1 AND CHANGED_WHEN < to_date('2014-11-10 16:22:39', 'YYYY-MM-DD HH24:MI:SS')

MAX_TEXT
SELECT /* xxx */ * FROM items_test$2 WHERE INBOX_ID = 999 AND CHANGED_WHEN < to_date('2014-11-10 16:22:46', 'YYYY-MM-DD HH24:MI:SS')

Gefunden!

Über 200.000 Varianten von:

zwei Literale mit
200.000 Kombinationen

```
SELECT *
  FROM items
 WHERE INBOX_ID = 1
   AND CHANGED_WHEN < to_date('2014-11-10 16:22:39',
                              'YYYY-MM-DD HH24:MI:SS');
```

Jedes für sich völlig unaufällig, aber alle mit der gleichen mit FORCE_MATCHING_SIGNATURE!

Der Übeltäter

Ein Deamon zum Auffinden von verwaisten Vorgängen.

Alles was länger als 2 Stunden nicht bearbeitet wurde wird verteilt.

- 3.500 Postkörbe
- alle 10 Minuten
- 24 Stunden am Tag

Deamon „Original“

```
FOR rec IN (SELECT INBOX_ID FROM inbox WHERE INBOX_ID != 0)
 LOOP
     l_sql := ' SELECT /* xxx */ * FROM items'
           || ' WHERE INBOX_ID = ' || rec.inbox_id
           || '   AND CHANGED_WHEN < to_date('''
           || to_char(SYSDATE - 2 / 24, 'YYYY-MM-DD HH24:MI:SS')
           || ''', ''YYYY-MM-DD HH24:MI:SS'')';
     OPEN crs FOR l_sql;
     LOOP
         FETCH crs INTO l_item_rec;
         EXIT WHEN crs%NOTFOUND;
         -- Process row
     END LOOP;
     CLOSE crs;
 END LOOP;
```

zwei Schleifen, zwei variable Literale!

Ein schlechtes Programm reicht!

- 1 Select je Postkorb = 3.500 Selects
- 6 mal pro Stunde
- 24 Stunden am Tag

3.500 * 6 * 24 = 504.000 Executions pro Tag:

= 504.000 neue Queries pro Tag

= 504.000 Hard Parses pro Tag

= 504.000 Busy Loops pro Tag (= CPU Waits)

Deamon done right!

```
FOR rec IN (SELECT i.item_id
              FROM inbox b
              JOIN items i
                ON i.inbox_id = b.inbox_id
             WHERE b.inbox_id != 0
               AND i.changed_when
                                < SYSDATE - 2 / 24)
LOOP
     -- Process row
END LOOP;
```

eine Schleife, statische Literale!

Guter Stil zahlt sich aus!

- 1 Select je Lauf
  mit **einem Hard Parse** beim Start der Datenbank.
- 6 mal pro Stunde
- 24 Stunden am Tag

1 * 6 * 24 = 144 Executions pro Tag:

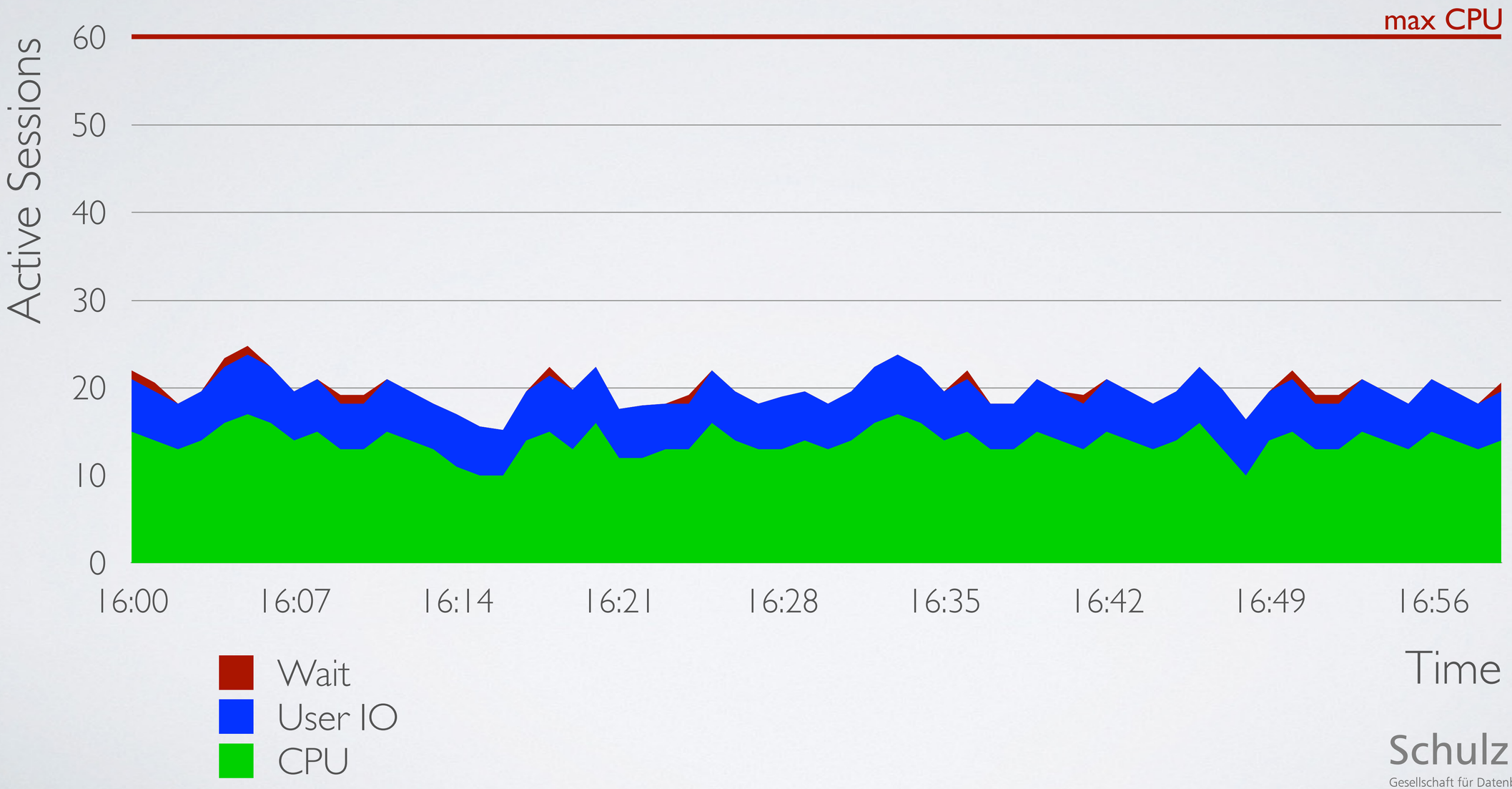
= 0 neue Queries pro Tag

= 0 Hard Parses pro Tag

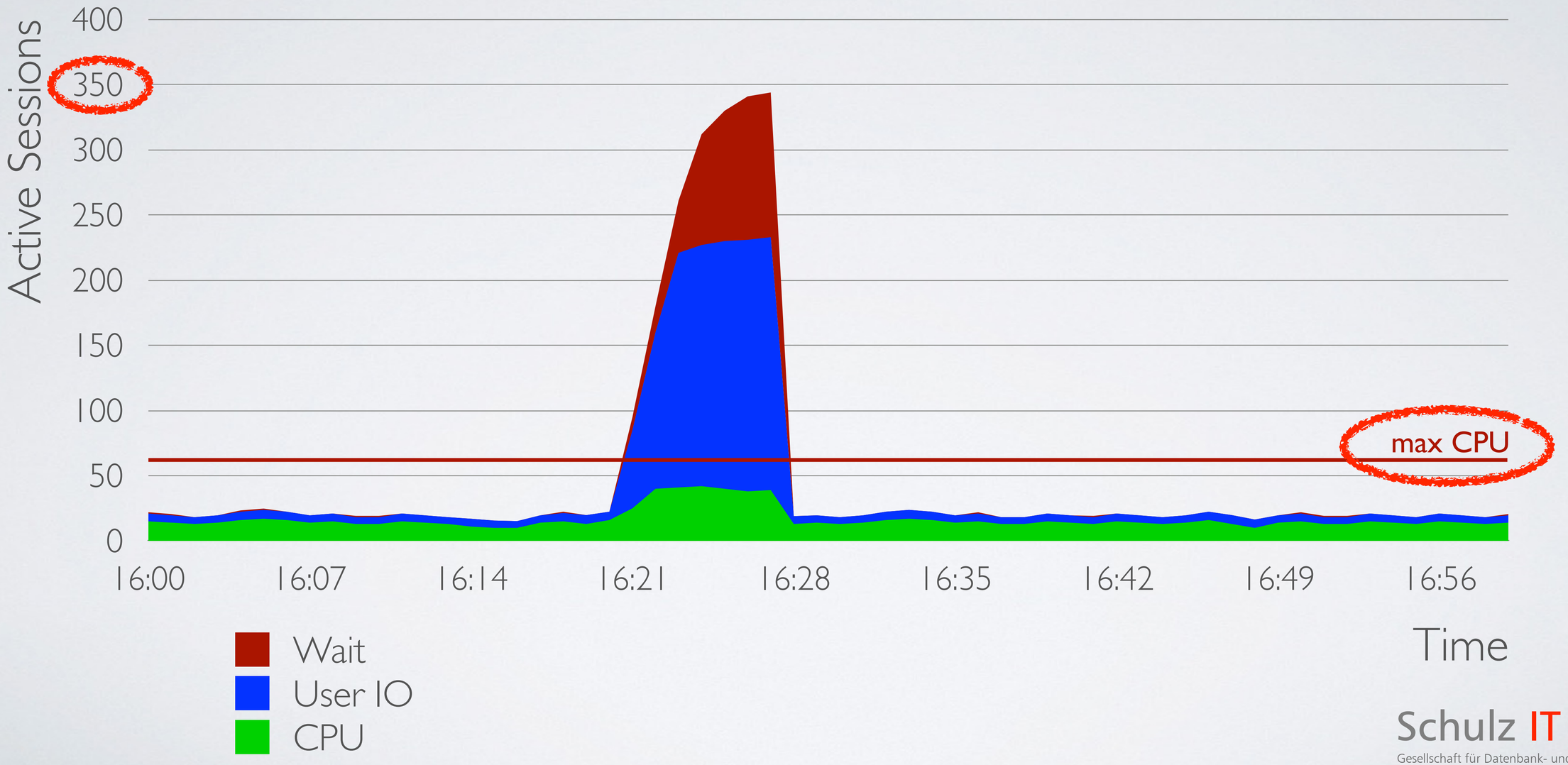
= 0 Busy Loops pro Tag

Einer für alle?

normale 16 Uhr Last



heute wird's später 16 Uhr Last



Ein SQL - zwei Aufgaben

Abarbeitung der neuen Dokumente nach dem FiFo Prinzip:

```
   SELECT d.*
     FROM document$ d
    WHERE d.inbox_id = :p_inbox_id
      AND d.document_id > :p_document_id
ORDER BY d.document_id;
```

Variable **`p_document_id`**:

1. zu Beginn der Bearbeitung ist **`:p_document_id`** = 0
2. während der Bearbeitung hat **`:p_document_id`**
   die ID des zuletzt bearbeiteten Dokuments (z.B. 18001100)

Normalbetrieb

Normalerweise entsteht der Ausführungsplan zum **Start der Bearbeitung** mit **`:p_document_id = 0`**

```
-------------------------------------------------------------------------------------
| Id  | Operation                    | Name               | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                    |     1 |    29 |     5  (20)|
|   1 |  SORT ORDER BY               |                    |     1 |    29 |     5  (20)|
|*  2 |   TABLE ACCESS BY INDEX ROWID| DOCUMENT$          |     1 |    29 |     4   (0)|
|*  3 |    INDEX RANGE SCAN          | DOCUMENT_INBOX_FK  |     1 |       |     3   (0)|
-------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - filter("D"."DOCUMENT_ID">:p_document_id)
   3 - access("D"."INBOX_ID"=:p_inbox_id)

Statistics
-----------------------------------------------------------
          4  consistent gets

Elapsed 0:00:00.05
```

Störung

Ein neuer Ausführungsplan ensteht **während der Bearbeitung** mit **`:p_document_id = 18001100`**

```
-------------------------------------------------------------------------------
| Id  | Operation                    | Name          | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |               |     1 |    29 |     4   (0)|
|*  1 |  TABLE ACCESS BY INDEX ROWID| DOCUMENT$     |     1 |    29 |     4   (0)|
|*  2 |   INDEX RANGE SCAN           | DOCUMENT$_PK |     1 |       |     3   (0)|
-------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter("D"."INBOX_ID"=:p_inbox_id)
   2 - access("D"."DOCUMENT_ID">:p_document_id)

Statistics
-----------------------------------------------------------
         15  consistent gets

Elapsed 0:00:00.01
```

Problem

Der neue Ausführungsplan wird nun auch für den **Start der Bearbeitung** verwendet mit **`:p_document_id = 0`**

```
-------------------------------------------------------------------------------
| Id  | Operation                    | Name          | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |               |     1 |    29 |   228K  (1)|
|*  1 |  TABLE ACCESS BY INDEX ROWID| DOCUMENT$     |     1 |    29 |   228K  (1)|
|*  2 |   INDEX RANGE SCAN           | DOCUMENT$_PK |   170M|       | 674220  (1)|
-------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter("D"."INBOX_ID"=:p_inbox_id)
   2 - access("D"."DOCUMENT_ID">:p_document_id)

Statistics
----------------------------------------------------------
     1730250  consistent gets

Elapsed 0:01:05.02
```

Lösung: zwei Abfragen - stabile Pläne!

- Start der Bearbeitung:

```
SELECT d.*
  FROM document$ d
 WHERE d.inbox_id = :p_inbox_id
ORDER BY d.document_id;
```

- Während der Bearbeitung:

```
SELECT d.*
  FROM document$ d
 WHERE d.inbox_id = :p_inbox_id
   AND d.document_id > :p_last_document_id
ORDER BY d.document_id;
```

Egoisten unter sich

ETL Lade-Prozess

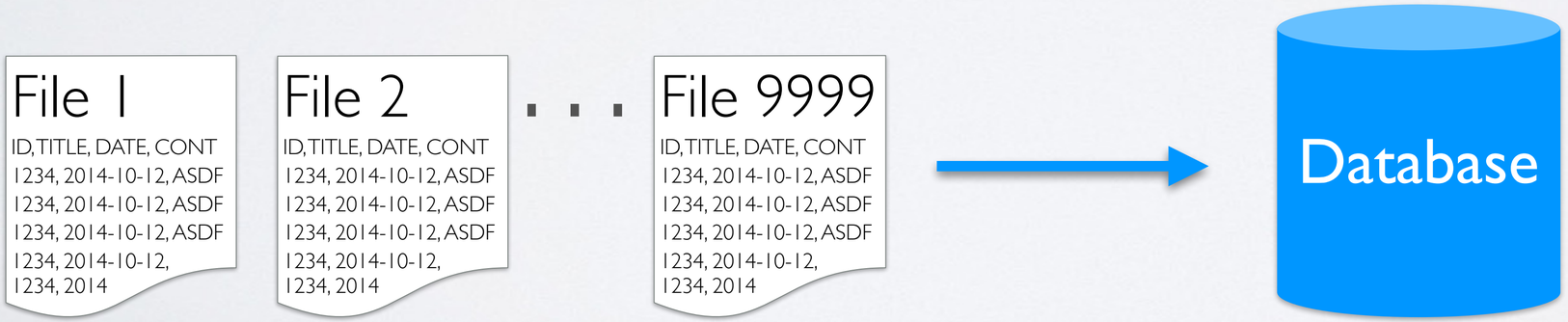
Die Aufgabe:

Laden von 1,5 Terra Byte in mehr als 1.000 Dateien.

Laufzeit 10 Stunden

Die Herausforderung:

Reduktion der Laufzeit!

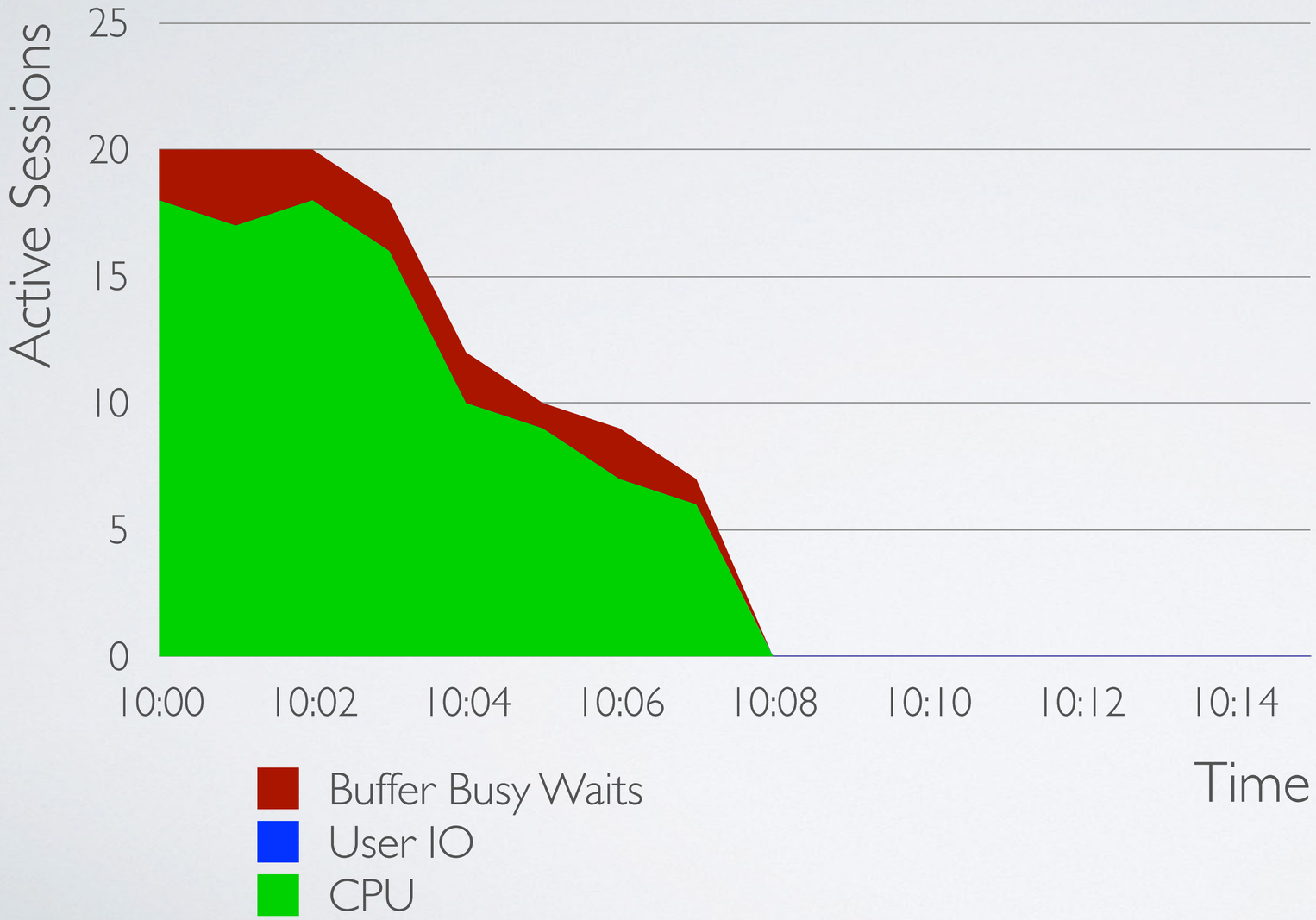


Insert

```
INSERT INTO APPEND_TEST$(ID, VAL)
SELECT LEVEL, RPAD('x', 4000, 'x')
  FROM DUAL CONNECT BY LEVEL < 1000001;
```

- 1 Million rows, simuliert mit „CONNECT BY“
- RPAD zur Erzeugen von Datenvolumen (400 Zeichen pro Zeile)

Insert



20 Millionen Zeilen

- 20 Tasks (gleichzeitig)
- je 1 Million Zeilen
- **Laufzeit: 7:01** Minuten

Oracle® Database
SQL Language Reference

APPEND Hint

In direct-path INSERT, data is appended to the end of the table, rather than using existing space currently allocated to the table.

As a result, direct-path INSERT ***can be considerably faster*** than conventional INSERT.

http://docs.oracle.com/cd/E11882_01/server.112/e41084/sql_elements006.htm#sthref496

Insert vs Insert Append

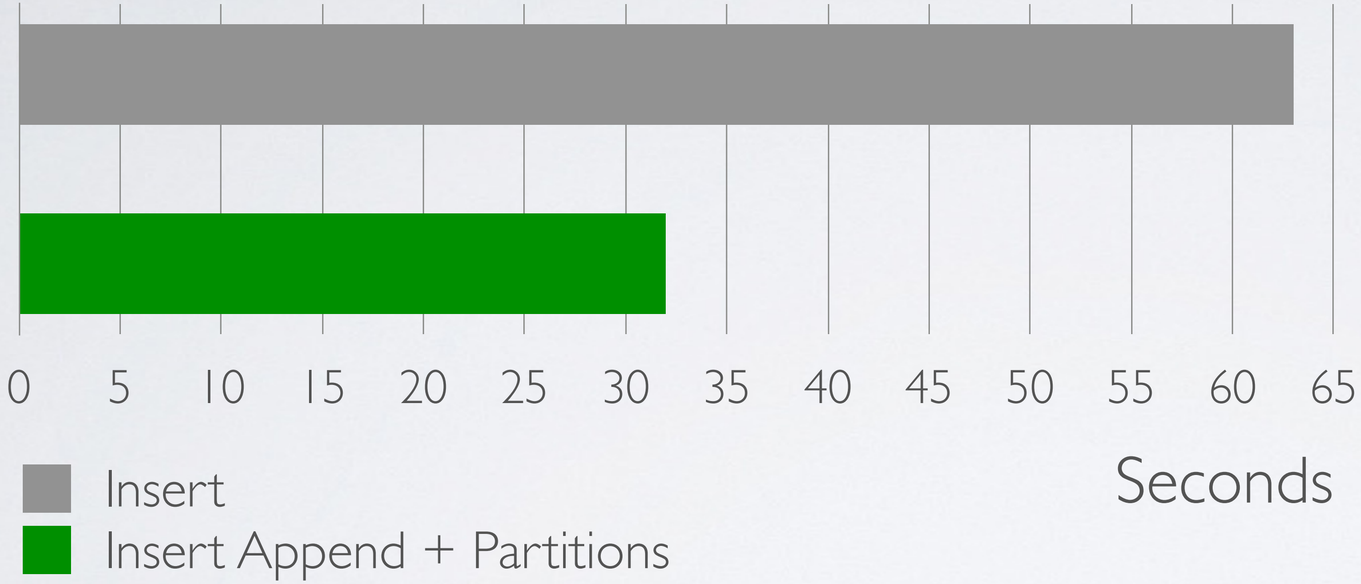
- ```
  INSERT INTO APPEND_TEST$(ID, VAL)
  SELECT LEVEL, RPAD('x', 4000, 'x')
    FROM DUAL CONNECT BY LEVEL < 1000001;
  ```

- ```
  INSERT /*+APPEND*/ INTO APPEND_TEST$(ID, VAL)
  SELECT LEVEL, RPAD('x', 4000, 'x')
    FROM DUAL CONNECT BY LEVEL < 1000001;
  ```

- 1 Million rows, simuliert mit „CONNECT BY“
- RPAD zur Erzeugen von Datenvolumen (400 Zeichen pro Zeile)

Insert vs Insert Append

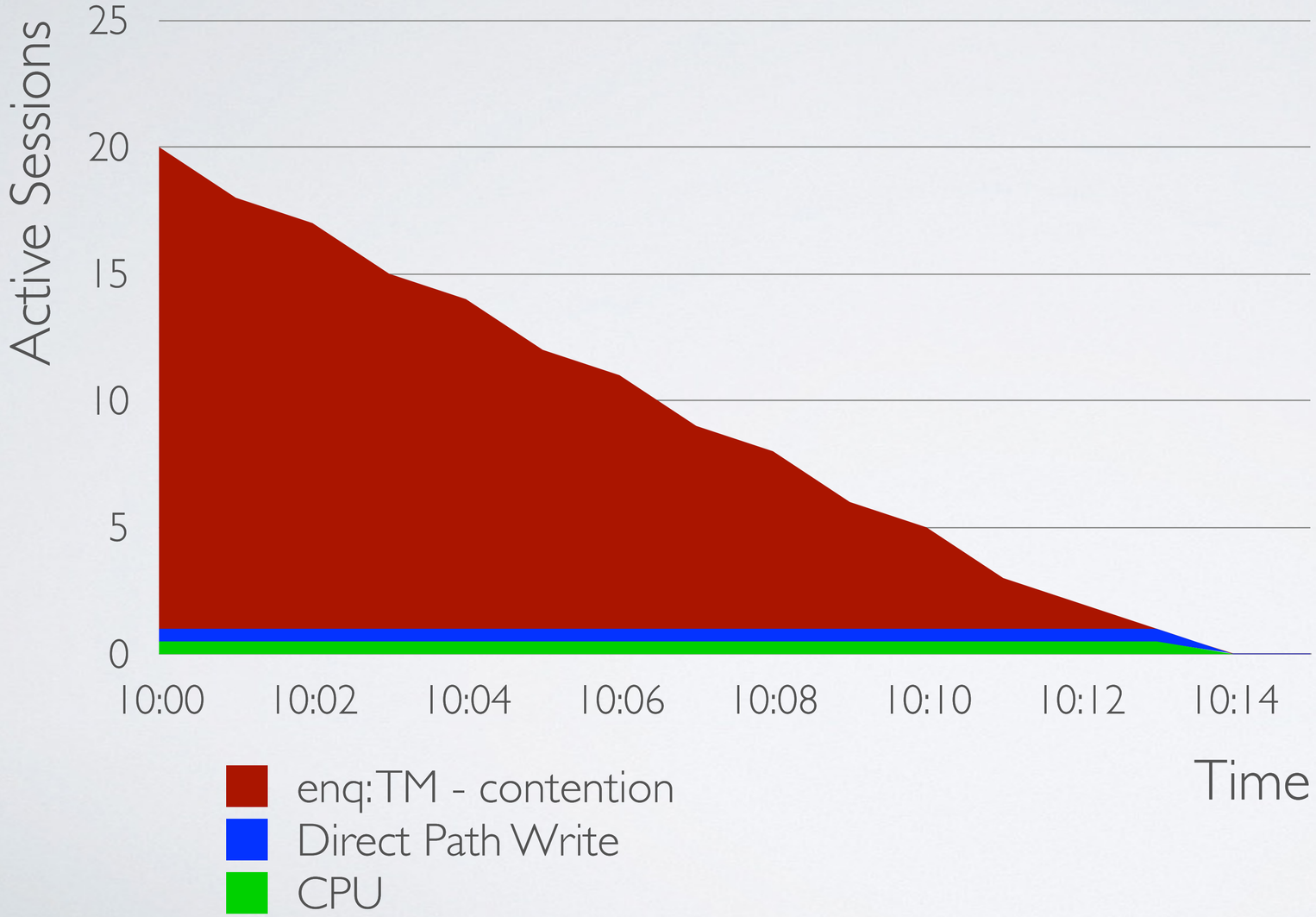
Test: Insert 1 Million Zeilen



Laufzeit:

- 63 s mit Insert
- 32 s mit Insert Append

Insert Append



20 Millionen Zeilen

- 20 Tasks (gleichzeitig)
- je 1 Million Zeilen
- \+ **Hint:** /*+APPEND*/
- **Laufzeit:** **13:07** Minuten

Oracle® Database Administrator's Guide

Locking Considerations with Direct-Path INSERT

During direct-path INSERT, the database obtains **exclusive locks on the table (or on all partitions of a partitioned table)**.

As a result, **users cannot perform any concurrent insert, update, or delete operations on the table,** and concurrent index creation and build operations are not permitted.

http://docs.oracle.com/cd/E11882_01/server.112/e25494/tables.htm#ADMIN015

Staging Tabelle bisher

```
CREATE TABLE append_test_part$
   (ID NUMBER, VAL VARCHAR2(4000));
```

Staging Tabelle mit 20 Partitionen

```
CREATE TABLE append_test_part$
   ( WORKER_ID NUMBER, ID NUMBER, VAL VARCHAR2(4000))
```

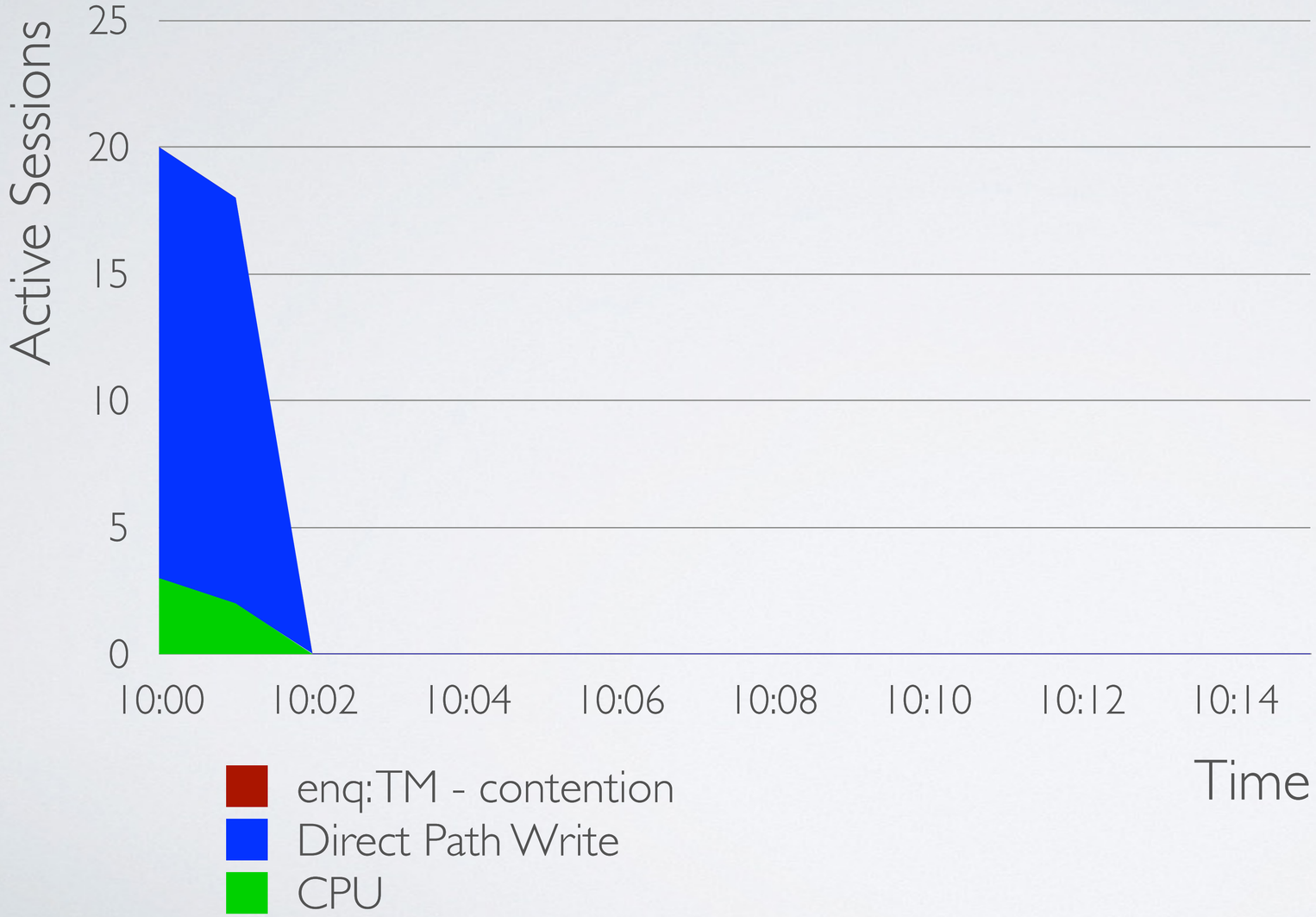
```
PARTITION BY LIST (worker_id)
(
    PARTITION w1  VALUES(1),    PARTITION w2  VALUES(2),
    PARTITION w3  VALUES(3),    PARTITION w4  VALUES(4),
    PARTITION w5  VALUES(5),    PARTITION w6  VALUES(6),
    PARTITION w7  VALUES(7),    PARTITION w8  VALUES(8),
    PARTITION w9  VALUES(9),    PARTITION w10 VALUES(10),
    PARTITION w11 VALUES(11),   PARTITION w12 VALUES(12),
    PARTITION w13 VALUES(13),   PARTITION w14 VALUES(14),
    PARTITION w15 VALUES(15),   PARTITION w16 VALUES(16),
    PARTITION w17 VALUES(17),   PARTITION w18 VALUES(18),
    PARTITION w19 VALUES(19),   PARTITION w20 VALUES(20)
);
```

Insert Append

```
INSERT /*+APPEND*/
  INTO APPEND_TEST$ PARTITION FOR(14)
       (ID,     VAL,                                WORKER_ID )
SELECT LEVEL, RPAD('x', 4000, 'x'), 14
  FROM DUAL
CONNECT BY LEVEL < 1000001;
```

- 1 Million rows, simuliert mit „CONNECT BY“
- RPAD zur Erzeugen von Datenvolumen (400 Zeichen pro Zeile)
- ***14*** muss mit der ID des jeweiligen Tasks ersetzt werden (hier mit Werten von 1 bis 20)

Insert Append + Partitions



20 Millionen Zeilen

- 20 Tasks (gleichzeitig)
- je 1 Million Zeilen

\+ Hint: /*+APPEND*/

+ 20 Partitions

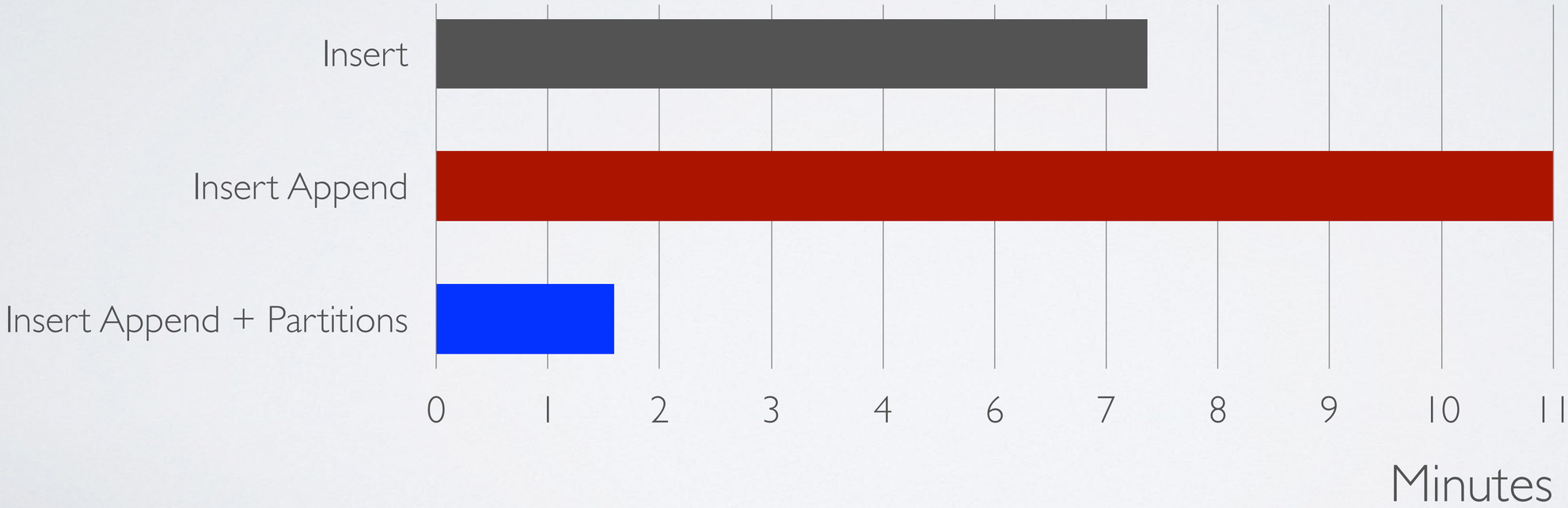
+ PARTITION FOR (value)

- **Laufzeit: 1:46** Minuten

Insert Varianten

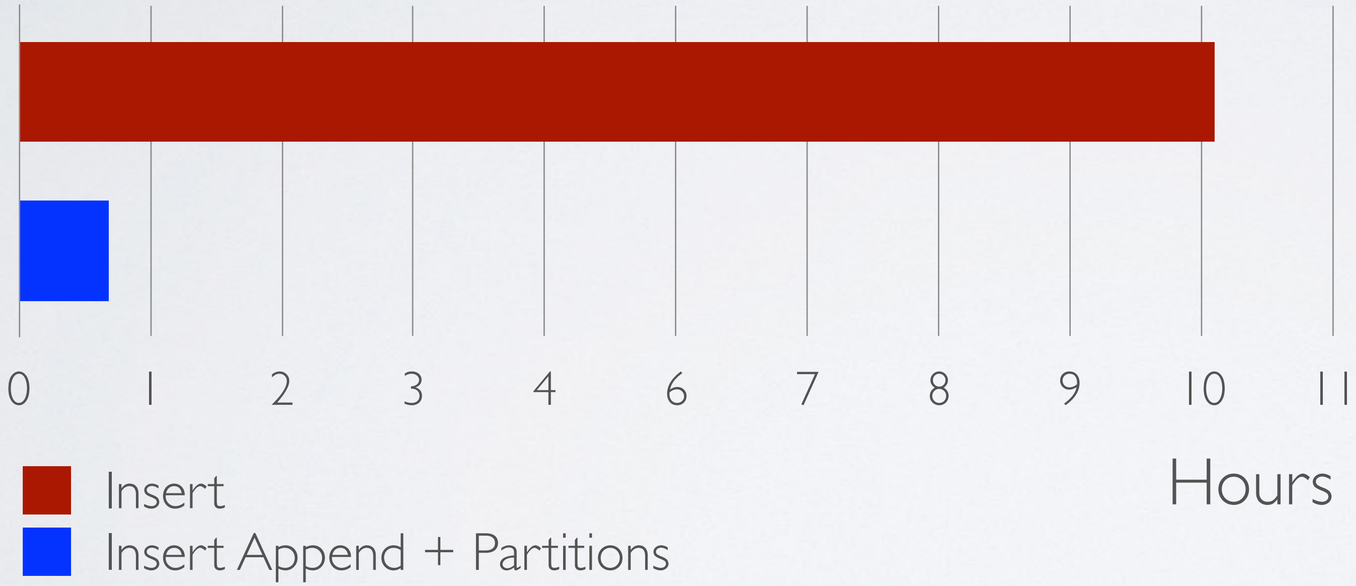
20 Millionen Zeilen

- 20 Tasks (gleichzeitig)
- je 1 Million Zeilen



ETL Lade-Prozess

Laden von 1,5 Terra Byte in mehr als 1.000 Dateien.



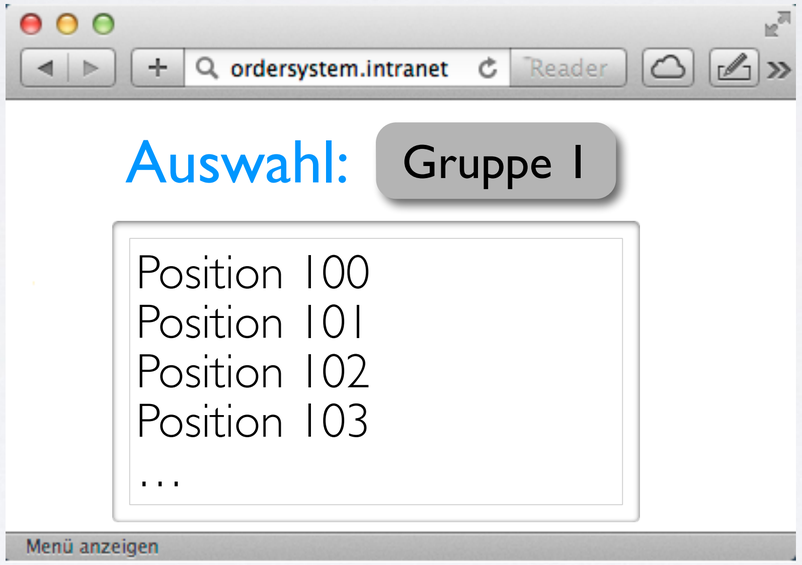
Laufzeit:

- vorher: 10 Stunden
- jetzt: unter 1 Stunde

1000 mal schneller

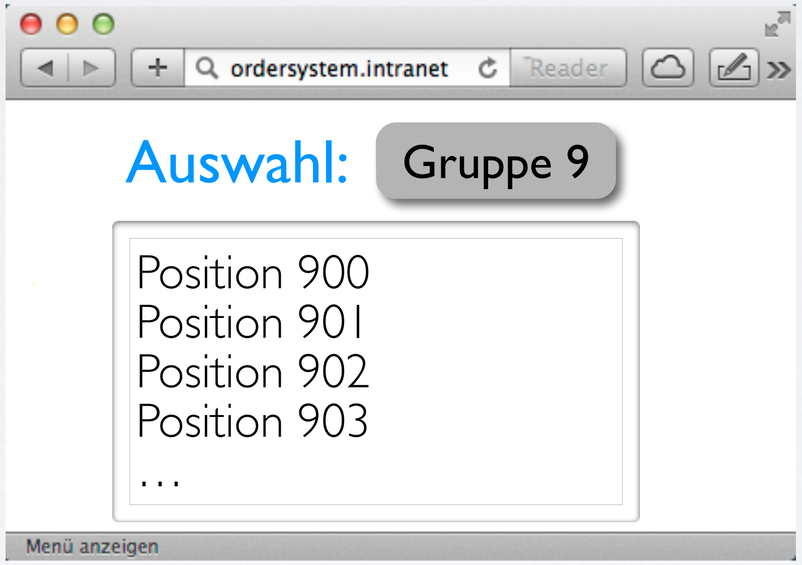
Anwenderoberfläche

Anzeige der Trefferliste nach Auswahl durch den Anwender



Das Problem

Die Anzeigendauer steigt plötzlich auf über 30 Sekunden!



Analyse der Abfrage

```
   SELECT pc.parent_id
     FROM parent_child_test$ pc
    WHERE pc.PARENT_ID BETWEEN 100 AND 200
 GROUP BY pc.parent_id;
```

- **parent_child_test$** ein View

Warum wird das langsamer?

Prüfen wir, wie viele Zeilen gelesen werden:

```
SELECT count(*)
  FROM parent_child_test$ pc
 WHERE pc.PARENT_ID BETWEEN 100 AND 200;
-> 101.000 rows
```

```
SELECT count(*)
  FROM parent_child_test$ pc
 WHERE pc.PARENT_ID BETWEEN 900 AND 1000;
-> 30.300.000 rows
```

Im zweiten Fall werden **300 mal mehr Zeilen** gelesen!

Analyse des verwendeten Views

```
CREATE VIEW parent_child_test$
AS SELECT p.parent_id, p.parent_val,
          c.child_id,  c.child_val
     FROM parent_test$ p
     JOIN child_test$ c
       ON (p.parent_id = c.parent_id);
```

- Es werden nur Spalten der Tabelle **`parent_test$`** verwendet.
- Die Tabelle **`child_test$`** hat keine filternde Wirkung (Vater-Kind-Beziehung).
- **Fazit:** Durch den View wird ein unnötiger Join zur Tabelle **`child_test$`** ausgeführt.

Weglassen des unnötigen Joins!

```
SELECT count(*)
  FROM parent_test$ pc
 WHERE pc.PARENT_ID BETWEEN 900 AND 1000;
-> 101 rows
```

Es werden nun **1.000 mal weniger Zeilen** gelesen!

```
SELECT pc.parent_id
  FROM parent_test$ pc
 WHERE pc.PARENT_ID BETWEEN 100 AND 200;
-> 101 rows
```

Es werden nun **300.000 mal weniger Zeilen** gelesen!

1000 mal schneller

Achten Sie auf die Verwendung ungeeigneter Views!

Schnell genug reicht nicht!
Oft zeigen sich die Probleme erst bei steigenden Datenmengen.

1000 mal schneller?

PARENT_ID	Laufzeit vorher	Laufzeit jetzt	Faktor (schneller)
100-200	0,19 s	0,01 s	**19**
900-1000	36,38 s	0,01 s	**3.638**

Fragen?

Haftungsausschluss

Die Schulz IT Services GmbH übernimmt keine Haftung für Vollständigkeit, Aktualität und inhaltliche Richtigkeit der zur Verfügung gestellten Informationen. Ebenso wird keine Haftung für Fehler redaktioneller und technischer Art übernommen.

Haftungsansprüche aus materiellen und immateriellen Schäden, welche sich durch die Nutzung fehlerhafter oder unvollständiger Informationen oder durch die Nutzung der angebotenen Informationen ergeben könnten, sind grundsätzlich ausgeschlossen, es sei denn, die Schulz IT Services GmbH trifft Vorsatz oder grob fahrlässiges Verschulden.

Sollten Formulierungen im Haftungsausschluss oder Teile davon mit der geltenden Rechtslage nicht mehr oder nur noch partiell übereinstimmen, so bleibt hiervon die Gültigkeit der übrigen Teile dieses Textes mit seinem Inhalt unberührt.

Urheberrecht

Danke!